Supporting Intellectual Work through Rendering and Review

Lars Yde

11 April 2001

Contents

1 Introduction and background **5**

1.1 Acknowledgements . . . 7

2 Purpose, methodology and overview **8**

2.1 Introduction . . . 8

2.2 The purpose of PeerView . . . 8

2.3 Methodology . . . 10

2.4 Supporting Intellectual Work Through Artifact Rendering and Group Review . 11

2.5 Overview . . . 12

2.6 Description of the Prototype . . . 13

2.6.1 System Overview . . . 13

2.6.2 Architecture . . . 13

2.6.3 The PeerView Client . . . 13

2.6.4 The PeerView Server . . . 15

2.6.5 Design Rationale . . . 15

2.7 Implementation Overview . . . 16

2.7.1 The User Interface . . . 16

2.7.2 Communication and Data Distribution . . . 16

2.8 Potential Uses for PeerView . . . 17

2.9 Earlier Work . . . 18

2.10 Future Plans . . . 19

2.11 Conclusion . . . 20

3 PeerView - background and details **21**

3.1 Analysis . . . 21

3.2 Design . . . 25

3.2.1 Interface design . . . 25

3.2.2 Functionality design . . . 27

3.2.3 Class design . . . 28
3.3 Implementation . . . 29
3.3.1 The scalable desktop . . . 30
3.3.2 Message and progress bar . . . 32
3.3.3 Communications layer . . . 33
3.3.4 Group directory . . . 34
3.3.5 Discussion forum . . . 35
3.3.6 Artifact updating . . . 35

4 Experimental evaluation of PeerView 39
4.1 Introduction . . . 39
4.2 The experiment . . . 39
4.3 Discussion . . . 40
4.3.1 Experimental findings . . . 41
4.4 User evaluation . . . 43

5 Implications of PeerView 1.0 for future systems 44
5.1 Introduction . . . 44
5.2 Design analysis . . . 44
5.2.1 Introduction . . . 44
5.2.2 Interface . . . 45
5.2.3 Functionality . . . 46
5.3 Implementation analysis . . . 47
5.3.1 Introduction . . . 47
5.3.2 Implementation language and technologies . . . 47
5.3.3 Representation of properties and messages . . . 49
5.3.4 Implementation of the distributed architecture . . . 50

6 InSiter 52
6.1 Introduction . . . 52
6.2 User audience . . . 52
6.3 Design objectives . . . 52
6.4 Interface design . . . 54
6.4.1 Introduction . . . 54
6.4.2 Screen designs . . . 55
6.4.3 Design rationale . . . 60
6.5 Architecture . . . 63

6.5.1 Information architecture . . . 64
6.5.2 Distributed architecture . . . 69
6.6 System requirements . . . 69

7 Conclusion **71**
7.1 Results and contributions . . . 71
7.2 Lessons learned . . . 72
7.3 Future work . . . 73

A Design documentation **74**
A.1 Use cases . . . 74
A.1.1 Add documents . . . 74
A.1.2 Remove documents . . . 75
A.1.3 Open group directory dialog box . . . 76
A.1.4 Update group directory . . . 76
A.1.5 Submit group directory . . . 77
A.1.6 Create group . . . 78
A.1.7 Delete group . . . 78
A.1.8 Edit group . . . 79
A.1.9 Join group . . . 79
A.1.10 Select document from visible documents box . . . 79
A.1.11 Zoom to overview . . . 80
A.1.12 Centre document . . . 80
A.2 Class diagrams . . . 80
A.3 Class diagram for package `clientapp` . . . 81
A.4 Class diagram for package `serverapp` . . . 82
A.5 Class diagram for package `peerviewmisc` . . . 82

B Evaluation experiment materials **83**
B.1 Task lists . . . 84
B.1.1 Danish version . . . 84
B.1.2 English version of task list for rank-and-file members . . . 88
B.1.3 English version of task list for group creator . . . 90
B.2 Transscript . . . 92
B.2.1 Danish version . . . 92
B.2.2 English version . . . 96
B.3 Questionnaires . . . 100

B.3.1 Danish version 100
B.3.2 English version of PeerView questionnaire for first participant 107
B.3.3 English version of PeerView questionnaire for second participant 108
B.3.4 English version of PeerView questionnaire for third participant 109

C Source code **112**
C.1 Introduction 112
C.2 List of listings 112

D PeerView help system **481**
D.1 Introduction 482
D.2 Add documents 483
D.3 Remove documents 485
D.4 Panorama 486
D.5 Discussion forum 489
D.6 Group directory 491
D.7 Preferences 493
D.8 Using a divider bar 496

Chapter 1

Introduction and background

Software development is difficult. This observation comes not only from my personal experience, but seems to be universally acknowledged by practitioners and researchers in the field alike [11, 51, 19]. Nor is it a particularly novel or sensational insight: numerous manuscripts and much research have recognized it, addressed it and, in many cases, proffered some panacea for remedying it. The problem persists, though, with no signs of abating. Software systems continue to be late, continue to frustrate their intended users and continue to be flawed or even dysfunctional [21]. The most important reason for what seems to be a perennial problem is probably the inherent complexity of software development with its complicated interplay of technological, social and psychological factors. The difficulty in managing this complexity is severely compounded by the intangibility of software engineering's subject matter. Contrary to other branches of engineering which are typically concerned with the transformation of physical matter in one form or another, software engineering is concerned with the creation of symbolic artifacts. In other words, software engineers have no immediate sensory perception of the product they fashion because it is, in essence, a purely intellectual construct. In his much quoted 1987 article [11], Fred Brooks states it so:

> "The reality of software is not inherently embedded in space. Hence, it has no ready geometric representation in the way land has maps, silicon chips have diagrams, computers have connectivity schematics ... In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable, and thus do not permit the mind to use some of its most powerful conceptual tools. This lack not only impedes the process of design within one mind, it severely hinders communication among minds."

Instead of direct observation, we rely on tools (compilers, debuggers, code analyzers, benchmarks, etc.) to help us gauge and probe the end product of a software development process, namely the code itself[1]. Such external measurement and experimentation can only process the syntactic and (formal) semantic properties of code. It does not yield information about its "sanity", that is, whether it is sensible relative to its problem domain or whether it makes

[1] A similar argument can be made for other phases, such as design and analysis but I will focus on program code for now.

logical sense[2]. To that end, a developer must still rely on human review conducted by herself or by others, either informally or systematized as code inspection meetings, walkthroughs or the like [40]. The value of review and scrutiny of program code and other symbolic artifacts, particularly by others than the author herself, should not be underestimated. Mathematicians have known and applied this knowledge for centuries through the institutionalized practice of proving the logical consistency of their ideas and then subjecting those proofs to scrutiny by other mathematicians. Software engineering has rediscovered the value of this mechanism and has accumulated statistical evidence to support it. For example, in [30] Humphrey writes that

> " ...if you were to inspect a software product that contained 100 defects, how many would you expect to find? We will call the percentage of the defects found the yield of the review or inspection. Again, there are no published data but my experience at IBM was that inspections typically yielded between 60 percent to 80 percent. Data from another organization support this with a reported 68% yield for one large operating system."

These characteristics of being intangible and subject to improvement through review is of course not exclusive to software. All artifacts of intellectual work seem to share them, consider for example coauthoring on scientific articles, proof-reading by editors, paper submissions by students to teachers, and so on. Software development is, however, conducted on an industrial scale and produces systems whose complexity dwarfs that of most other human artifacts. It might therefore be the area that could benefit the most from technologies, for example in the form of software tools, that could make software artifacts perceptible to its developers and thus more susceptible to both traditional and entirely new forms of inspection and review as the artifacts were being developed. Ideally, such tools would render a representation of a set of artifacts and the process by which they are brought about and would provide facilities for the generation and accumulation of review material concerning that representation. That is, in its fully fledged incarnation, the support tool envisioned would provide a both spatial and temporal rendition of a software project and facilities for communicating about it.

In Chapters 2 and 3 of this thesis, I present and discuss a prototype for such a tool, called PeerView, which was implemented by me in the latter half of 2000. The first part of that presentation will be an overview of this prototype in the form of an article coauthored with Jyrki Katajainen, which appeared as [62] and has been submitted to an international journal[3]. It is followed by a more thorough discussion of the material it introduces. In Chapter 5, I analyze the design, implementation and development process of the PeerView project, and discuss what lessons can be derived from it. Next, in Chapter 6, I apply those lessons to the design of a more ambitious system that is to succeed and in part build on the PeerView prototype. This system, which has been given the working title *InSiter*, will be implemented if the necessary resources can be made available, and will hopefully then be subjected to extensive usability testing. The thesis concludes with a brief summary and look at results achieved and experience garnered as well as a perspective on the future of this work.

[2]The "sanity" of a design, a piece of code or other symbolic information might be called its *natural semantics* to distinguish it from formalized descriptions of its semantics.

[3]Another, somewhat shorter article appeared in the February issue of Distributed Systems Online [61]

1.1 Acknowledgements

Thanks to my supervisor Jyrki Katajainen who provided guidance and ample room for discussion, both technical and otherwise, and who helped ensure that i's were dotted and t's crossed by proof-reading the draft manuscript. Also, through financial support from grant 9801749 (project Performance Engineering), he made it possible for me to focus on thesis writing, not making ends meet. Thanks also to Morten Nielsen who gave useful feedback on late drafts of the manuscript which helped improve the final version. The three participants in the experimental evaluation — Paul Broholm Abrahamsen, Thomas Hyldahl and Rikke Iversen — provided valuable oral and written feedback by allowing me to tape record our communication during the experiment and by subsequently completing a questionnaire. I would like to thank them also for their assistance. The Department of Computing at the University of Copenhagen (DIKU) proved helpful by providing me with office facilities and funds for external consultancy on portions of the thesis manuscript. Thanks also to Kasper Hornbæk for some interesting feedback on the PeerView interface. Finally, thanks to my girlfriend Jeanette Vedelstjerne Gersager who helped me regain what I was at times lacking in moral fibre.

Errors, omissions and typos are hopefully few and far between, but whatever may persist are of course entirely of my doing.

Chapter 2

Purpose, methodology and overview

2.1 Introduction

This Chapter provides an overview of the prototype software that was developed for this thesis and the problems that it addresses. This Chapter also includes a description of the work methodology used for the present thesis, i.e. what was the intended progression of events from inception to conclusion. The concluding Chapter (Chapter 7) assesses how this methodology was implemented, i.e. how was the correlation between what was intended and what actually happened.

2.2 The purpose of PeerView

PeerView is an exploratory prototype. Its purpose is to provide groups of users involved in intellectual work with a *common information space* and facilities for communicating about the contents of that information space. The notion of a common information space (CIS) is not unequivocally defined in existing literature, but in [4], Bannon and Bødker discuss the concept and give a description that can be summed up for the purposes of this work by the following quote from [48]:

> " ...a central archive of organizational information with some level of 'shared' agreement as to the meaning of this information (locally constructed), despite the marked differences concerning the origins and context of these information items."

They go on to note that:

> "Cooperative work is not facilitated simply by the provision of a shared database, but requires the active construction by the participants of a common information space where the meanings of the shared objects are debated and resolved, at least locally and temporarily."

PeerView is primarily intended as a CIS system for software developers since the artifacts of their shared work are often voluminous and it can be difficult to formulate the mental

equivalent of the physical representation that manual workers have of their work simply by virtue of their artifacts being physical. In other words, if one's subject matter are strings of symbols, the very act of making sense of what one is working on takes effort, whereas with physical matter, the act of making sense is an unconscious "making of sense" through processing of sensory data that does not require the sometimes painstaking intellectual effort needed to understand complex symbolic representation. Common information spaces thus seem to derive their justification from the way in which they can help automate the work of constructing and maintaining representations of external activities, namely shared work. The potential benefits are not only the savings in work time and effort that one would suspect to see when individual collaborators can more quickly and systematically access and absorb the information needed to carry out their work, but also the organizational rewards that can potentially be reaped from an integrative model of shared work. By the latter I mean that when workers and management can access a comprehensive and coherent model of the work that goes on within their organization or group, they acquire an object of both conversation and contention. They can discuss and monitor progress, either informally or by means provided by the workspace system itself, and can review the shared artifacts themselves in a more immediate way than if they were not accessible at a central locus. It seems plausible that this can ultimately affect not only the work process itself, but also the organization within which that process is undertaken since such a model can give all stakeholders in the organization a common focal point for discussion. The above is largely speculative and the intention behind PeerView is not only to give users a support tool for their collaborative work, but also to explore how these ideas can best be translated into reality.

In [27], Gutwin and Greenberg refer to a particular type of common information space, namely the *shared worksspace* as "a bounded space where people can see and manipulate artifacts related to their activities." [27, Section 2.2]. They go on to note that "In these spaces, the focus of the activity is on the task artifacts: the visible and manipulable objects through the task is carried out." [27, Section 2.3]. This form of common information space is limited in that it usually seeks to reproduce or simulate a physical workspace such as a desktop or a conference room. Other forms of shared information, such as organizational structure or awareness of where coworkers are physically located and what they are doing, need not be present in the shared workspace or even deducible from it. If it is, the relationship between workspace artifacts and such awareness information might be entirely coincidental. PeerView's dominant feature is such a shared workspace where users can share and manipulate artifacts. The purpose of providing this workspace is both to give users a window onto the work of whatever group they are currently associated with and to experiment with the design and implementation of this type of common information space. The latter has yielded useful information, both through the process of development itself and through experimental evaluation, as discussed in Chapters 4 and 5, respectively. This information can be fed into the development of other systems that use common information spaces to support collaboration, one of which is outlined in Chapter 6.

In addition, PeerView demonstrates novel methods of navigation, at least in a groupware context, by giving users a fully zoomable desktop on which the shared artifacts can be placed and manipulated. This is an experimental way of addressing the "detail/overview" problem of how to represent both local detail and global overview in a usable manner which is discussed in [22], where the authors describe a different solution to this problem, namely the fisheye lens. PeerView also addresses the practical problem of catering to groups operating in a

heterogeneous computing environment by being implemented in pure Java, i.e. without any native code, and therefore able to run on a variety of machine architectures and platforms.

Being a prototype, PeerView also helps answer the development question: how can a fully fledged CIS support system best be developed? PeerView may be limited in its functionality but it incorporates all the components that should be present in more ambitious systems requiring many man-years of development time, and the experience gained during all phases of PeerView development can facilitate implementation of such systems. Chapters 3 and 5 describe the development process and discuss what lessons can be learned from it.

PeerView imposes little structure on its users and their work process. This means that it affords users a great degree of freedom which is in keeping with an observation by Grintner [23]: "one of the emergent trends in workflow research ... [is] to make the systems more flexible to accomodate the contingent aspects of work.". However, it also means that PeerView does little to systematize and automate work for its users. Later systems, such as that described in Chapter 6, will have to address the specific needs of a more narrowly defined user audience to go beyond the simple functionality found in PeerView. In a sense, this implies a range of possible applications from the pure, unadorned support of shared information in a CIS to a highly structured, user customizable workflow system centered around a CIS. Viewed in that light, PeerView is clearly towards the former end of the spectrum. For that same reason, PeerView may appeal to a broad audience but does not purport to appeal to an equally broad range of work situations. That is, although PeerView may be used by anyone from the office clerk with basic computer literacy to the veteran software developer, it does not necessarily support all of their work situations and nor does it have to in order to fulfill its purpose.

2.3 Methodology

The methodology used in this project is a hybrid resulting from the need to reconcile scientific exploration with the practicalities of software development. The idea for a distributed application supporting intellectual work through rendering and review sprang from reading [52] among other works, and from personal experience with software development projects where lack of communication and coordination can complicate or derail efforts. Now, this conception of an idea could have been followed by lengthy elaboration process where the initial ideas were translated into a detailed paper model, i.e. a verbal and diagrammatic description, and not necessarily implemented as an executable application. This approach would obviously have allowed for a much more ambitious initial design since the lenghty implementation cycle of "code-execute-debug" could have been eliminated. However, I believed such an approach to be of questionable scientific value at best and worthless at worst since a pure "armchair design" that was not put to an experimental test through implementation would be much like a physics theory that was never empirically tested, i.e. insubstantial though possibly intellectually stimulating. Also, most software development methodologies advocate incremental or iterative development [6, 42, 32] which corresponds to my personal experience which is that most system development benefit from many short iterations rather than monolithic, phased design as in the traditional "waterfall" model [51, p. 9]. I therefore decided to advance quickly from idea to design and implementation, followed by *a posteriori* analysis of the development process and evaluation of the finished prototype in the hope that this would yield valuable feedback based on empirical data that could help improve the design and development of

future systems.

Putting a methodology on a summary form as done below is slightly misguided as methodology is more a mindset and structured approach than a formulaic procedure. However, it is illustrative to see the overall steps taken put into a list form so that is done here:

1. Inception of PeerView
2. Elaboration of PeerView
3. Construction of PeerView
4. Transition
5. Evaluation
6. A posteriori analysis
7. Elaboration of successor system (next iteration)

To anyone familiar with the unified software development process [32], the above sequence will be recognizable, at least in part, because the activities "Inception", "Elaboration", "Construction" and "Transition" are identical to the phases in a "workflow cycle" to use unified process parlance. However, this is not in itself of scientific merit since these developmental phases do not include formal evaluation of the end product and the process by which it was brought about, and therefore does little to verify their validity. The evaluation and analysis phases of the process used for this work are intended to address this shortcoming and help give this thesis the scientific potency needed to have an impact beyond the software product resulting from it. In Chapter 7, I discuss how the actual progression of events corresponds with the process intended and what room there might be for improvement.

2.4 Supporting Intellectual Work Through Artifact Rendering and Group Review

The following is a slightly abbreviated version of an article that appeared as [62] and has been submitted to an international journal. Only the introduction has been cut to avoid overlap with Section 1. The article appears here to provide an overview of the PeerView software.

Supporting Intellectual Work through Artifact Rendering and Group Review

Lars Yde
Department of Computing, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
larsyde@diku.dk

Jyrki Katajainen
Department of Computing, University of Copenhagen
Universitetsparken 1, DK-2100 Copenhagen East, Denmark
jyrki@diku.dk

Abstract. Intellectual teamwork, such as that done by software development teams, is characterized by the intangibility of its subject matter. This can make it difficult for team members and outside stakeholders to gain an overview of the product being built. Many software tools of various designs have addressed this problem, often by using graphs and charts to model ongoing projects. In this paper, we suggest a design based on artifact rendering and group review and argue that it can promote overview and communication. We also present software built from that design and describe its potential uses and audience as well as its implementation.

Keywords.Computer-supported collaborative work, component-based development

Category.H.4.3, H.5.3, K.4.3

2.5 Overview

In [Section 2.6] we describe a prototype of a tool intended to support *intellectual teamwork* in general. That is, we aim at supporting any process in which a symbolic product, an *artifact*, is being produced by individuals in group collaboration, so the use is not limited to software developers. Our main design goal was to promote overview by artifact rendering and communication by group review. By *artifact rendering* we mean the representation of intellectual artifacts by some medium to facilitate processing by humans, and by *group review* the activity through which team members exchange information and commentary on the artifact being built. Our prototype uses visualization to render an artifact that consists of textual and graphical *documents*, but other forms can be envisaged, i.e. auditory representation. Also, discussion fora can be associated with parts of the artifact rendered.

Our prototype was developed using the Java programming language and a set of compatible component technologies. In [Section 2.7] we discuss both and argue that implementation was possible only through such component-based development, given our limited resources. Next, in [Section 2.8], we propose some potential uses of the prototype, both in its present form and in subsequent versions. We then go on to discuss earlier related work in [Section 2.9] and finally, in [Section 2.10], we outline our plans for a more advanced system.

2.6 Description of the Prototype

The prototype aimed at fulfilling the basic needs of artifact rendering and group review was named *PeerView* and work on it was begun in the middle of May 2000 as part of Lars Yde's M. Sc. thesis. At the time of writing (mid October), a beta release has been completed and a stable version is planned for release before the end of the year. The system is released under a freeware license both in its binary and source form. In this section, we outline PeerView's design.

2.6.1 System Overview

PeerView provides users with a dynamic representation of on-going work by rendering the artifact of that work in a form which allows easy overview as well as inspection and discussion of detail. PeerView accomplishes this by allowing groups of users to share documents and have them rendered in a scalable visual panorama. The set of documents is constantly kept updated so as to provide a real-time window onto the artifact (e.g., a collection of code files) being developed. With each document is associated a discussion forum that enables a group to have discussions on the document in a manner similar to a USENET [57] newsgroup having discussions of a set topic.

2.6.2 Architecture

PeerView is a client/server application with "fat" clients and a "thin" server, meaning that most system functionality is placed in the client application and relatively little in the server. Each PeerView user is assumed to be running a single instance of the client program on his or her machine, but multiple instances are possible, provided they are running from separate locations (different directories). The server may run on one of the users' machines or it may run on a separate host. Currently, PeerView supports two types of connection: TCP/IP sockets and HTTP. The former is intended for local or closed networks, the latter for use over the Internet, but both may of course be used differently.

2.6.3 The PeerView Client

The client application provides a window onto a set of documents via a zoomable *panorama* [Fig. 2.1, centre] where documents can be arranged in an arbitrary pattern and zoomed to arbitrary scale as well as moved and sized individually. By centring a document, the user can access its *discussion forum* [Fig. 2.1, bottom] and read as well as add contributions to the on-going discussion on that document. The top of the client window is occupied by a selection panel consisting of standard *drop-down menus* and *toolbar buttons*. The drop-down box in the right-hand side of the toolbar panel lists the documents currently displayed in the panorama so that each can be selected by name as well as by navigating the panorama using a mouse.

From the selection panel, the user can choose to add or remove documents, customize the client using a preferences dialog or access and modify the *group directory*, i.e., the list of groups currently available. The directory appears on top of the main window as shown in [Fig. 2.2] where the panorama is zoomed to overview, displaying in this case 32 documents arranged

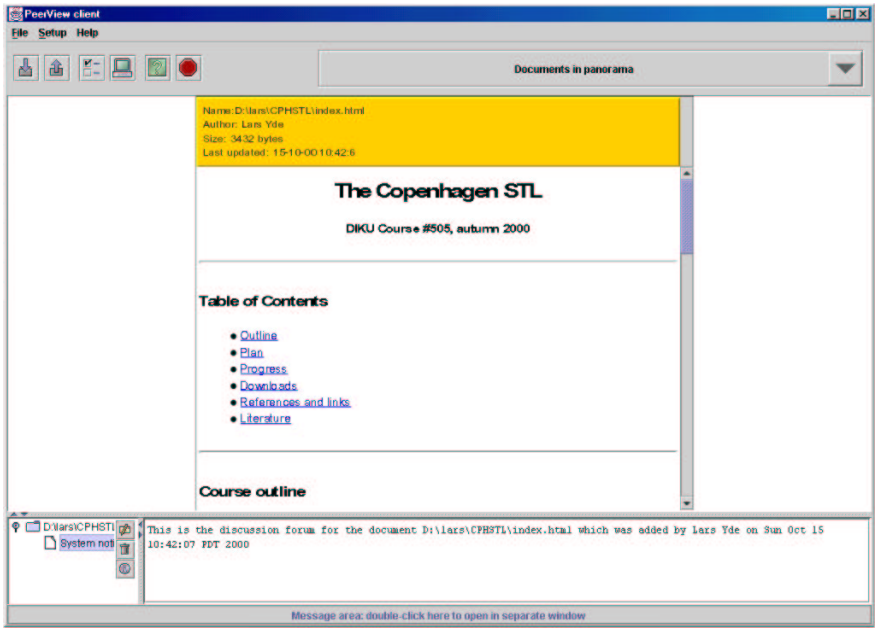


Figure 2.1: The PeerView client window.

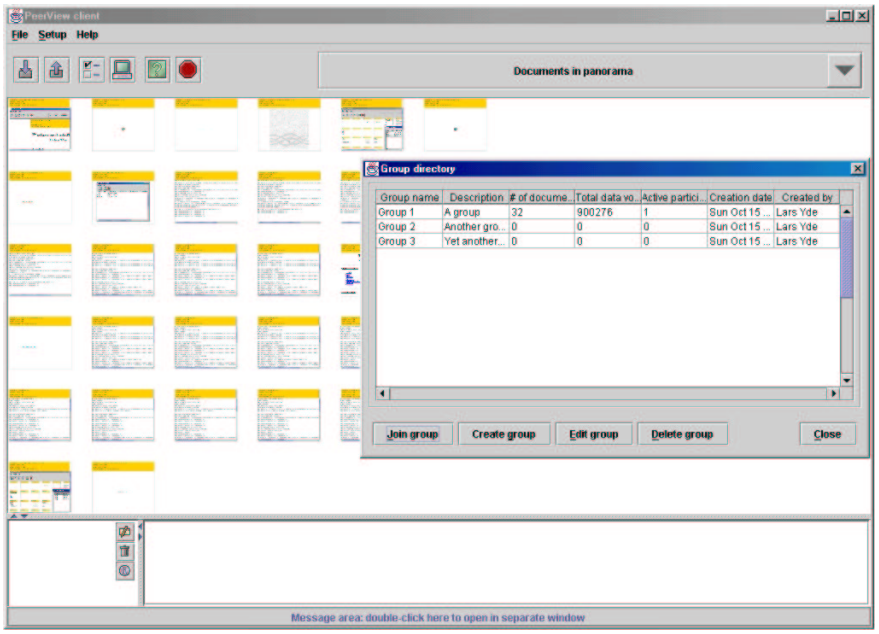


Figure 2.2: Panorama overview and group directory.

in a grid pattern which can be customized by the user from the preferences dialog available from the drop-down menus. Each group is listed with its current number of participants, the number of documents it comprises and their total volume in bytes. Using the button panel at the bottom of the group directory box, the user can join, create, edit and delete groups. After joining a group, the other group members, if any, are instructed by the client application to submit their documents so that they can be added to the panorama of the new group member.

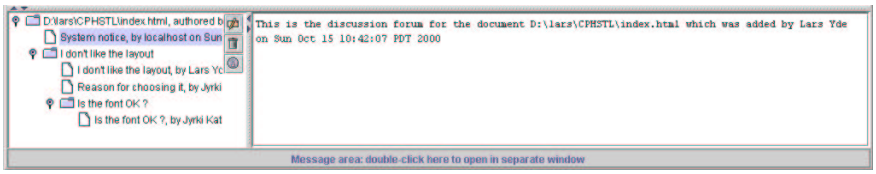


Figure 2.3: Discussion forum.

The new member's own documents are then broadcast to the other group members so that their panoramas can be similarly updated.

As indicated earlier, centring a document (by double-clicking it) causes the PeerView client to request the discussion forum for that document from the server and, upon receipt, display it in the bottommost section of the main window. From there, the user can select a position in the *topic tree* [Fig. 2.3, left] and using the small vertical toolbar panel choose to compose a new message, delete an existing message that has not yet been responded to, or view the *property sheet* of the selected message.

2.6.4 The PeerView Server

The central purpose of the server is to maintain a database of clients, groups and discussion fora which is reflected in its user interface [see Fig. 2.4]. It consists of little else than a listing of the groups supported, a toolbar and a menu panel from which a few, basic functions (setup and help) can be selected.

2.6.5 Design Rationale

The choice of a client/server architecture was made based on the responsibilities of the server. A peer-to-peer architecture would have been equally feasible but would merely have placed the server functionality in an arbitrary client, resulting in little discernible difference to the user. More significant is the use of a zoomable panorama. As explained earlier, the underlying motivation is to provide overview without loss of content. The user can achieve this by zooming the arrangement of documents to his or her preferred scale, of course, but by combining zooming with a linear setup, an effect similar to that found in work by Eick et al. on the visualization

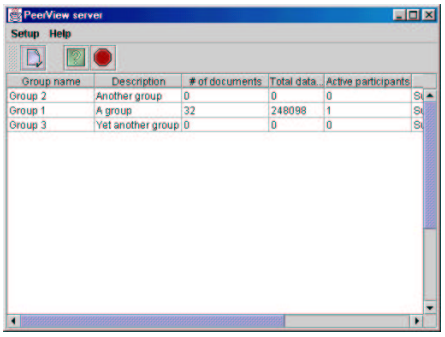


Figure 2.4: The PeerView server window.

tool SeeSoft [52, p. 315] can be produced. SeeSoft uses a linear arrangement of multi-coloured columns to render an overview of a collection of code files and their associated statistics (such as time of modification), thus giving users access to large amounts of information at a glance.

On a more theoretical level, the idea of a customizable panorama that can be viewed at arbitrary levels of detail is in keeping with the notion of micro/macro readings as described by Tufte in [55, p. 38 ff.]:

> "...the power of micro/macro designs holds for every type of data display as well as for topographic views and landscape panoramas. Such designs can report immense detail, organizing complexity through multiple and (often) hierarchical layers of contextual reading."

Tufte's focus is on the presentation of cartographical and scientific data but his observations seem to apply equally well here.

2.7 Implementation Overview

The most important considerations in choosing implementation technology for PeerView were: broad user appeal, ease-of-use, high degree of portability, extensible structure and reliability. The Java programming language seemed the obvious choice given these requirements, but implementing all of the planned functionality bottom-up within the set time-frame of four months was almost surely infeasible since implementation was in the hands of a single developer (Lars Yde). The solution lay in using component libraries to implement some of the low-level features such as data distribution and visualization that would otherwise have demanded weeks of development, testing, and fine-tuning to function satisfactorily. The specific choice of components and the reason for choosing them are detailed below.

2.7.1 The User Interface

The PeerView interface is implemented using the Swing classes found in the Java Development Kit (version 1.2.2) [50]. These are light-weight graphical components meaning that they make no (or only very limited) use of operating-system-specific code, thus ensuring maximum portability and a minimum of visual variability across platforms. The zoomable panorama which occupies the centre portion of the client window is built using the Jazz toolkit (version 1.0) [34], which provides primitives for building scenegraphs and scaling text and/or graphics. (A *scenegraph* is a tree structure for organizing graphical objects.) Jazz interacts smoothly with other Java technologies and its structure facilitates future extensions and modifications. In that sense, it allows development time to be shortened without imposing arbitrary restrictions on future development efforts.

2.7.2 Communication and Data Distribution

The PeerView communication infrastructure is implemented using the Java Shared Data Toolkit (JSDT, version 2.0) [36]. It is a relatively low-level toolkit compared to some of the alternatives surveyed, such as NCSA Habanero [28], TANGO Interactive [53], and DreamTeam

[18], all available online and free of charge. However, it has the distinct advantage of being 100% pure Java, meaning that the risk of incompatibility problems — a frequent stumbling block to developers — seemed minimized.

The JSDT requires a server to maintain a directory of clients and groups, and allows all members of a group to communicate and exchange data using an arbitrary number of *channels*. Currently, PeerView uses only a single channel per group, but extending the design later on would be easy as would adding a number of advanced features such as data encryption, access restriction and communication statistics gathering, all of which are supported in part or whole by the JSDT.

As mentioned earlier, PeerView currently supports two communication protocols, namely TCP/IP sockets and HTTP, but can easily be extended with additional protocols since the JSDT design is completely independent of its underlying implementation. Such an extension is, however, a low priority as other, more interesting features (suggested in the previous paragraph) are higher on the agenda.

2.8 Potential Uses for PeerView

PeerView was designed for a specific purpose, namely support of intellectual teamwork, but was deliberately made flexible through simple design and portable technology. It is therefore suited for other uses than that of facilitating intellectual work. Some possible applications of PeerView are listed below. Many of these are inspired by examples found at the Jazz website [34].

Repository inspection: PeerView could be used for the inspection of document repositories such as those maintained by the version management systems that are often used in software development. It is planned to try this in practice in the near future by allowing users to download the PeerView client software from a website and then access a CVS repository using that software (CVS is short for Concurrent Versions System — a popular open source version management system, see [14]).

Web browsing: PeerView can display HTML documents in its present form and can be extended to support hyperlinks in future versions, and could, after such an extension has been implemented, function as an inter-/intranet browser.

Presentation: Documents can be displayed and arranged in PeerView's panorama and subsequently used in a presentation much like conventional slides but in a more manageable way, for example, by projecting the panorama onto a canvas and then using a mouse to navigate while giving the presentation.

Authoring: PeerView could be used as an authoring tool by researchers who could have separate copies of a shared manuscript placed on the panorama and then edit and annotate their version while conferring with others about their copies. This could also be used for commentary and proof-reading and would scale well since the zoomable PeerView panorama need not consume more user screen space as the number of authors increases.

Education: Due to its simple design and graphical interface, PeerView is accessible to most users, including children and some groups of disabled people, and so could be used both for distance learning in geographically dispersed communities and as a support tool in conventional education. One obvious example of the latter is letting visually impaired students study teaching materials at their preferred magnification using the zoomable PeerView panorama.

The above points to some general areas of application, but the number of specific uses is potentially quite large because of PeerView's extensible, open-ended architecture and design.

2.9 Earlier Work

The extensive directory of computer-supported collaborative work and groupware resources at [56] lists a large number of groupware applications for shared editing, conferencing, e-mailing, and other forms of group collaboration. In theory, such functionality can serve the same purpose as PeerView, but in practice, many of these products are designed for different purposes and are thus ill suited for artifact rendering and group review. Two examples are Cybozu Office [15] and Lotus Notes [38], both of which are large suites of tools for collaborative work that facilitate resource sharing and communication. However, their size and complexity makes it impractical to use them for the narrowly defined and specific purpose that PeerView has.

Several academic initiatives have explored areas related to artifact rendering and group review and have produced software that addresses the same underlying problems as PeerView. Examples are the TeamRooms software [46] created for supporting team collaboration and group awareness, and the WORLDS software [60] used for distributed access to one or more repositories of information. Both are similar in architecture to PeerView and are aimed at facilitating collaborative work through object sharing and group communication, but their designs differ from PeerView's in key areas. Both TeamRooms and WORLDS are based on spatial metaphors, i.e., the notion of shared electronic spaces. TeamRooms realizes this idea by letting users define *rooms* which are persistent fora where objects can be shared and communication can take place between the visitors of the room. Each room is equipped with facilities for collaborative work in the form of applets for exchanging information and carrying out other tasks. The TeamRooms user interface is dominated by a panorama onto which the contents of the room (i.e., the shared objects) are projected for inspection and manipulation by all users "present" in the room. An implication of such a design is of course that the set of shared objects can easily occupy more than the visible screen space and TeamRooms addresses this problem by offering a room overview radar that gives an outline, non-scalable bird's eye view of the room (the work on radar overview has been continued at the University of Calgary [24] where TeamRooms was developed).

On the conceptual level, PeerView differs from TeamRooms by being a tool for artifact rendering and group review whereas Teamrooms provides shared electronic spaces for collaborative work. This is the central difference since the conceptual nature of a system usually dictates future design and development efforts. In terms of design, PeerView has fewer features than TeamRooms and consequently a simpler interface. It also addresses the issue of artifact overview differently, namely through its scalable, configurable panorama, but it is similar in its

use of object sharing by projection onto a panorama surface. Technologically, the most important difference is that PeerView is written in Java while TeamRooms is a Tcl/Tk application which obviously impacts portability and potential audience since Java is platform-independent and Tcl/Tk is not, although it is widely supported. The WORLDS collaboration environment, which is called *Orbit*, differs from PeerView in much the same way as TeamRooms, but has a more complex architecture and, consequently, more stringent system requirements than both TeamRooms and PeerView.

2.10 Future Plans

Given the open design and implementation of PeerView, we expect little difficulty in integrating new features suggested by future users and ourselves. A handful of useful extensions have already been brought to our attention and are listed below:

- New display managers to allow advanced document formats such as RCS (the format used by CVS [14] for storing the revision history) and streaming video in addition to the existing formats which include plain text, HTML, RTF, GIF and JPG.
- A selection of layout managers, i.e., user configurable program components for automated control of the documents in the client panorama, in addition to the default manager (the grid layout manager) now available.
- An editable and extensible property sheet associated with each document.
- Discussion fora for groups of documents organized by owner, topic, or some other shared characteristic.

The above features are for integration into future releases along with any additional suggestions deemed worthwhile.

The long term plan is to gather feedback about the use of PeerView and from that assess the viability of its philosophy and design. Provided this assessment is favourable, we plan to develop a more ambitious system for real-time artifact rendering and *process rendering*. That is, in addition to an artifact, the organization and history of a project is rendered. The system is to assemble global statistics about the state of a project by collating information and documents gathered from individual clients distributed in a network. Relevant statistics would be the number of documents worked on by each user, the number and position of changes made to those documents and, for code files, profiling and debugging information. That information should then be organized by the server and transmitted around the network to an arbitrary number of consumers who would render it, either visually or by some other medium determined by the preferences of the user. Moreover, the system could offer the review facilities of PeerView along with a set of more advanced features. Ideally, the latter would include "scribbling pads" to communicate in pictures/diagrams in addition to words, audio and video conferencing to allow real-time communication among group members, and playback facilities for observing the evolution of a project over time. Thus, by combining artifact rendering with process rendering, a both spatially and temporally accurate representation of on-going work is given.

2.11 Conclusion

The problems that PeerView seeks to address are not new, but its approach and design are relatively untried. Only user feedback and continued research will tell if such an approach is justified and has enough substance to bear larger initiatives. If so, the potential benefits from a more ambitious project as outlined seem significant. On a more philosophical level, the potential merit of the PeerView approach comes from the fact that it is a response to a real and pervasive problem in software engineering, namely how to get a cognitive grip of an intrinsically intangible and usually highly complex subject matter. In that sense, PeerView can be seen as a small step in the evolution towards CASE tools that can help software engineering attain the maturity of other engineering disciplines and thereby, hopefully, stem the pandemic of failed projects.

Software Availability

The latest PeerView binaries are available online at:

`http://www.diku.dk/research-groups/performance-engineering/PeerView/`

The source code and accompanying documentation will be made available at the above address before the end of this year (2000). Future developments and release plans will also appear there as will related resources. Please feel free to download PeerView and test its usefulness for yourself.

Acknowledgements

This work was supported by the Danish Natural Science Research Council under contract 9801749 (project Performance Engineering).

Chapter 3

PeerView - background and details

The article format places some natural restrictions on style and content, so the level of detail in the above has by necessity been limited. The following paragraphs seek to remedy that by providing the background details on the rationale for the design of PeerView, the reasons for the choice of implementation technologies and architecture, and a discussion of the development process. In the below, PeerView is referred to with its current version number, 1.0, when necessary for the specific point being made or item being discussed. In all other cases, it is simply referred to as PeerView, i.e. without a version number.

3.1 Analysis

The analysis phase of the PeerView 1.0 design consisted mostly in refinement of the ideas that underpinned it, namely support for intellectual work through artifact rendering and group review. For a full-scale system, this would have included empirical research among a target audience to ascertain their preferences and proficiency. Even for a prototype system like PeerView 1.0, field research would have been useful, but I deemed it infeasible within my set timeframe. Naturally, that decision influenced my perspective on the design: PeerView was to be a vehicle for certain ideas, and at the same time be potentially useful to a wide audience to demonstrate the generality, in addition to the feasibility, of those ideas. The implication to me was that PeerView should be simple enough to maintain focus on the core ideas and facilitate wide use, yet comprehensive enough to provide both real functionality for its users and material for future research.

The central issues that had to be resolved first are discussed below in the order they were considered.

The semantics of representation

A central question was of course: what was meant by artifact rendering in practice? What needed clarification, in other words, were the semantics of representation, i.e. what constituted an artifact and which of its constituents should a representation render to provide a useful model for its users.

The answer lay in the fact that any artifact of intellectual work can be represented by a finite

string of symbols. The plausibility of this statement follows from its wording: if intellectual work has resulted in the creation of an artifact, then it has been converted from its abstract state ("thought stuff") to material form (a physical artifact). If we assume that all material objects can be exhaustively described using some symbolic formalism, then any artifact can be encoded in a computer readable form since any such formalism can be converted to binary numbers using a binary enumeration of its symbols. This is an abstract formulation, to be sure, but it allows us to focus on the essential aspect of computerized artifacts, namely content. Whether the artifact be text, graphics, audio, or something else entirely, it will always have a string representation. This is one defining property of an artifact, but it is so abstract a definition that it is almost vacuous. Any rendering mechanism would require additional information, i.e. "meta-data", about the artifact such as its medium (text, graphics, audio), its author, possibly its revision history and so on. The exact composition of the meta-data should depend on the type of rendering used: 3D graphics might require different information than 2D graphics, for example.

The form of representation

With a workable, albeit abstract, definition of the artifact concept in place, the next step was to decide on an appropriate form of representation, i.e. the appearance of the artifact rendition. I chose visual rendering as the basis for the interface design, but although this may seem the obvious choice it is not predetermined, at least not if one defines visualization as done in [52, p. 32]:

> "Visualization may be defined as 'the power or process of forming a mental picture or vision of something not actually present to the sight' ...Notice that this definition allows for the use of sensory modalities other than vision, e.g. hearing ...to assist in the formation of mental picture or images."

The reason for my preferring a visual rendering mechanism was the conceptual and technological difficulty I foresaw in implementing a system that supported multiple sensory modalities or merely relied on a single modality other than sight. Also, it seemed overly experimental to base a prototype design on something as relatively untried as auditory, tactile or, in the extreme, olfactory perception!

The visual interface should of course be tailored to needs and expectations of its intended audience, but there were other, more specific concerns. The design chosen would for example have to address two recurring problems for designers of graphical interfaces, namely visual clutter and overcrowdedness, meaning that interfaces can fill up with objects such as windows stacked upon each other until they make up an unmanageable jumble. For systems based on the desktop metaphors there are various solutions such as virtual desktops and multiple desktops on the same machine, but for systems with a potentially large number of objects, such as groupware applications relying on object sharing, this may not suffice. Such has been the experience of Greenberg et al. who in [47] describe the problem as follows:

> "A central concern in information visualisation is how a system can present both global structure (that provides overview and context) and local detail (that reveals information in the user's area of interest)."

They then propose two complementary software solutions to the problem, namely a fisheye view mechanism for use with shared editors so that multiple authors can follow each others work within the same focus, and a magnification lens for use with a graph editor, allowing customized magnification of portions of a graph being edited.

I did not become aware of the above research until after I had begun implementation, but I seem to have followed a similar line of reasoning, namely that the central problem was that of combining local display with global view. This initially led me to 3D graphics as that intuitively seemed the best way to represent a large volume of visual information within the confines of a desktop display. However, the advantages of 3D display, namely increased information volume, flexibility of design and navigation and appeal to human spatial cognition should be weighed against its disadvantages, summarized in [1]:

- Initial confusion and disorientation combined with the additional complexities of the interface and greater freedom of movement. Navigating with six degrees of freedom within an information space as opposed to navigating a set of 2D windows on an information space through panning and zoom controls.
- Large processing overheads are evident when a typically fast computer is reduced to crawl by a complex visualisation. Additional equipment such as a large high resolution monitor and specialized input devices are also a benefit, though by no means a necessity.
- Navigating any 3D environment requires a level of spatial awareness from the user. Navigating within an abstract information space demands more from this skill. Such spaces do not typically conform to our preconceptions of a 3D environment and various expectations such as a notion of "up" and "down" can easily be violated.

Also, it was my experience that 3D graphics displays are more taxing to develop than 2D graphics displays since the former is inherently more complex, thus requiring more resources to implement, debug and maintain. This gap has been closing for some time as freely available component libraries mature (such as OpenGL, see [44]) since these can help lift the burden of implementing a 3D renderer. The point remains, though, that 3D graphics add complexity to a user interface and have the added disadvantages listed above. I therefore decided to use 3D only if I could not formulate a satisfactory 2D alternative.

The challenge was therefore to create a 2D display that could be used for detailed inspection of each element in a set of artifacts yet easily transformed to provide an overview of arbitrary subsets. A scalable desktop seemed to have all the desired characteristics. If one had a 2D surface based on the standard desktop metaphor that could be scaled to arbitrary levels of magnification, one could place a set of artifacts on that desktop and navigate it along the x, y and z axes, thus solving the problem of visual clutter while adhering to GUI conventions known by most users and well supported technologically. In terms of implementation, this solution seemed manageable since scaling a desktop is equivalent to scaling a bitmap which is a well researched graphics transformation [2, p. 78 ff.]. This still left the exact appearance of each artifact on the desktop to be determined, but I considered that a design question and thus left it to be determined at that stage.

Modes of communication

To be a prototype for both the ideas of artifact rendering and group review, PeerView should provide facilities for communicating about its rendering in addition to facilities for displaying it. There are a variety of existing models for supporting group communication, such as mailing lists, USENET news, online chat, email services, and so forth. These provide suitable metaphors for communication facilities in some existing groupware systems such as FirstClass [12] and Lotus Notes [38], and were therefore both tried and conventionalized which made them seem safe, albeit conservative, choices of communication medium. I considered opting for a more experimental solution, but decided to err on the side of caution, bearing in mind the intended audience for PeerView and the time constraints I faced[1].

I reasoned that PeerView 1.0 should at least have facilities for threaded discussions in the manner of a USENET discussion [57] and, preferably, support for online peer-to-peer communication (chat) since this would allow review material to be accumulated and organized in a tried and tested form, and still allow sundry communications to take place more privately, i.e. without the perhaps dissuasive logging that discussion fora entail. Adding more features might contribute to usability, but only at the expense of interface simplicity and development time.

Architecture

Since PeerView was to be used for group review about a set of artifacts which, by implication, would have be shared, it was clear that the system architecture had to be distributed. There are a number of ways to implement such an architecture; some commonly used models are peer-to-peer, client/server and multicast, so a choice still had to be made between the options available. Under ideal circumstances, this choice would be made without consideration for the technologies available; one would simply assume that the required means of implementation were obtainable and design accordingly, i.e. without thought of technological limitations other than the state-of-the-art. However, this is obviously an unrealistic stance since lack of technology support can thwart even the best of designs. It was infeasible for me to build a distributed architecture bottom up within my set timeframe. I therefore decided to only survey component libraries for building communications layers in distributed applications, preferably tailored specifically for collaborative applications. This meant that the implementation language had to be decided on since any communications library was apt to be specifically targeted at a single programming language.

I considered `C++`, Java and SmallTalk as possible implementation languages because they seemed to offer the best mix of extensive library bases, cross-platform support, good development facilities and reasonable run-time performance. Java was chosen over `C++` and SmallTalk because of its inherent cross-platform support and the existence of suitable component libraries for the implementation of PeerView. In fact, the existence of suitable component libraries came to dictate the choice of programming language rather than vice versa. In particular, it quickly became apparent that numerous groupware toolkits [25] existed that were almost all implemented in Java and that more general purpose libraries for collaborative applications, notably the Java Shared Data Toolkit (JSDT) also were available. I chose the latter for PeerView

[1] Section 6 has a more detailed discussion on alternative implementations.

since the toolkits I surveyed all had drawbacks: either they were discontinued, as in the case of Habanero [28], not suited [18] or otherwise unsatisfactory [53]. Furthermore, JSDT had the added advantage of being pure Java and licensed by Sun which minimized the potential for development difficulty such as interfacing different technologies.

JSDT can be used to implement client/server and multicast architectures, but the server need only be assigned minimal responsibilities such as maintaining a registry of clients so it could be used to construct an architecture that effectively would function in a peer-to-peer manner. I was therefore not concerned about limiting myself to a particular architecture prematurely since JSDT seemed flexible enough to accomodate potential revisions later on in development.

3.2 Design

A commonly used distinction is that between the interface of an application, meaning the graphical components that the user interacts with, and the functionality, meaning the actions that the user can carry out by interacting with the interface. This is not the only possible division, of course, but it reflects the actual design process, so the below discussion reflects that view.

3.2.1 Interface design

Java was chosen as the implementation language for PeerView during the analysis phase, and this in part dictated the design of the GUI since the look-and-feel (as the graphical appearance and response pattern of an interface are sometimes called) of the interface widgets (i.e. windows, buttons, scrollbars etc.) should be that provided by the Java AWT or Swing interface classes. Choosing a different look-and-feel did not seem sensible as the Java classes implement all common interface primitives and can be extended arbitrarily. I chose the Swing class set which is Java's lightweight, i.e. platform independent, GUI library rather than the AWT which shipped with early releases of Java, but is now being phased out.

The specific layout of the PeerView interface should both cater to its intended audience by being accessible and intuitive or at least adhering to convention, yet at the same time fulfill its other purpose, namely that of clearly demonstrating the ideas it is built on. The most important design goal could therefore be stated as providing the user with access to the functionality required for carrying out a particular task without distracting from that task, preferably in an aesthetically pleasing manner. In practical terms, this meant sparse displays, i.e. windows and boxes which are not densely packed with widgets, and sensible layout, e.g. associating selection controls with the display elements they controlled. This is supported by Schneiderman [49, p. 318] who reports:

> "Crowded displays are more difficult to scan, especially for novice users. In a NASA study of space-shuttle displays, sparsely filled screens with approximately 70-percent blanks were searched in an average of 3.4 seconds, but more densely packed screens with approximately 30-percent blanks took an average of 5.0 seconds ... This study also demonstrated that functionally grouped displays yielded shorter search times."

It is also in keeping with the heuristics formulated by Nielsen in [43] who advocates:

> "The ideal is to present exactly the information the user needs — and no more — at exactly the time and place where it is needed. Information that will be used together should be displayed close together ..." [43, p. 116]

By aesthetically pleasing I did not mean merely according to my personal preferences, but in the sense of consistent use of simple and tasteful visual presentation to further usability. This issue is discussed at length by Nielsen who, among other things, offers the following aphoristic advice on colour, information density, and graphic design, respectively:

> "Don't overdo it. An interface should not look like an angry fruit salad of wildly contrasting, highly saturated colors. It is better to limit the design to a small number of consistently applied colors." [43, p. 119]

> "The 'less is more' rule does not just apply to the information contents of the screen but also to the choice of features and interaction mechanisms for a program. A common design pitfall is to believe that 'by providing lots of options and several ways of doing things, we can satisfy everybody'. Unfortunately, you do have to make the hard choices yourself." [43, p. 121]

> "Screen layouts should use the gestalt rules for human perception to increase the users' understanding of relationsships between the dialogue. These rules say that things are seen as belonging together, as a group, or as a unit, if they are close together, are enclosed by lines or boxes, move or change together, or look alike with respect to shape color, size or topography." [43, p. 117]

Having settled on the guidelines for design, I immediately began translating it into reality using a rapid prototyping tool. Some software engineering pundits recommend designing displays on paper before committing them to electronic form, but with the graphical design tools available today, I find this approach unwieldy. I started by designing the PeerView client main window since this was the "control center" of the application, so to speak. The result is shown in one of the figures used in chapter 2.4, namely Figure 2.1. The window is divided into three parts: the control section at the top, the middle portion containing the scalable desktop and a bottom portion containing the discussion forum, which in turn is divided into a navigation frame and a content frame. The sizes of these elements, except the control Section, can be adjusted using the divider bars[2] that separate them. This layout was chosen because it correlated with the conventions of GUI design by having control at the top, work area below and text input areas at the bottom. I would have preferred to have the quality of this design verified through suitability testing in a realistic setting, but had neither the time nor the means to do so, which was another reason for making conservative design choices. The drop-down menus and toolbar in the control Section were populated by items reflecting the functionality I could foresee from the results of my analysis, and were therefore likely to change as design, and perhaps even implementation, proceeded. The same applied to the pop-up menu that could be activated

[2] Vertical and horizontal bars with a ridged surface and arrow points at one end.

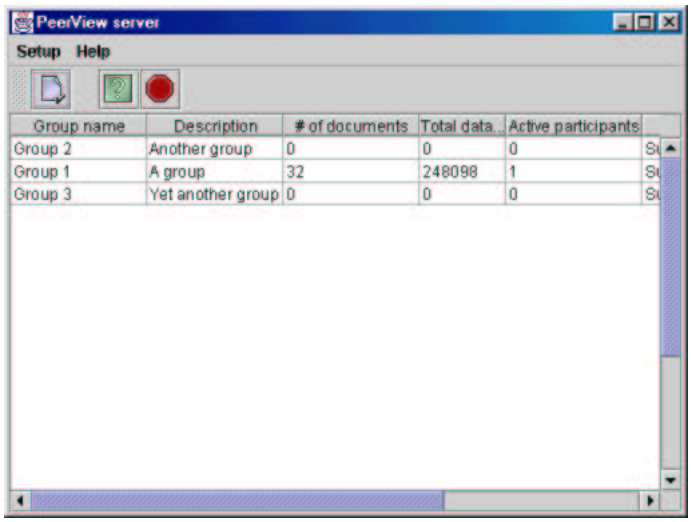


Figure 3.1: The PeerView server main window.

by right clicking on an artifact, i.e. I put in the menu items I believed necessary to have a functioning product in place and left room for extensions if need be. The list box in the right hand side of the toolbar contains a listing of artifacts, i.e. documents, shown in the panorama organized by author. By selecting a title from the box, the user can center the panorama on the corresponding artifact. This form of iterative development may seem meandering, but it is, in my opinion, more realistic than a rigid, linear design process that leaves little room for exploration.

The PeerView server window, shown in Figure 3.1, reflects its purpose, namely to serve as registry and nexus for communication between clients. Consequently, it is kept simple with only a control Section at its top and a listing of groups below that.

3.2.2 Functionality design

As I had found suitable technologies for implementation during the analysis phase, some of the design issues had already been settled implicitly. Specifically, the Java Shared Data Toolkit implements the concept of *clients* partaking in *sessions* using *channels* as lines of communication. Using that technology therefore meant designing functionality accordingly, i.e. supporting similar concepts. The PeerView client application was therefore designed to operate in groups, which were to be implemented as JSDT sessions ("group" was chosen because it connoted a more permanent affiliation than "session"), and communicate using JSDT channels. This was the bare minimum of communications infrastructure, but I decided to limit the initial design to that to keep implementation manageable although the JSDT contained other facilities that could have benefited the design.

The functionality of the scalable desktop was more of an open question as it depended largely on what could be implemented in the time available. The Jazz library [34] proved helpful in answering that question. Jazz is a library of classes specifically targeted at creating scalable (or zoomable, if you will) interfaces, appropriately named ZUIs, and by using that library, I could manage considerably more sophisticated designs than if I had had to implement everything

bottom up. Jazz provides primitives for constructing and managing a scenegraph [8] which is a data structure for organizing objects in a 2D or 3D environment and it implements a fast scaling transform for graphical objects. Furthermore, it can be used with the standard Swing document and bitmap classes and all in all, this seemed to promise a shortened development phase by allowing me to use component technology for both the scalable desktop itself and the artifacts to be rendered on it. In keeping with the conservative design choices made earlier, I decided to limit desktop functionality to the essential operations, namely translation[3] of individual artifacts, scaling of the entire desktop and automatic layout of all artifacts. Later, after having implemented and tested these features, I realized the need for additional ones, but given the pop-up menu interface to the desktop functionality, it was straightforward to add more features by simply implementing the required functions and adding items to the menu. Specifically, it proved useful to be able to "pin" artifacts to the desktop, i.e. fix their positions so that they were not repositioned once a rearrange operation was carried out, so this and a handful of other features were added to the desktop menu before releasing the beta version.

To put these ideas of functionality into more concrete form, I wrote a series of use cases, i.e. simple narratives describing typical scenarios of use, as suggested by [33]. Writing use cases after having outlined functionality and interface was somewhat unorthodox as standard procedure is to begin by writing use cases, preferably in collaboration with the intended users of the system. However, this was not a standard development project in that the problem domain and audience were abstract rather than concrete, so it did not make sense to me to follow standard procedure. That is, in the absence of a concrete problem domain, one has to flesh out one's ideas before use case narratives can be written about them. This could be either in the abstract, i.e. in the mind, or in more concrete form through design which is what I opted for in this case. Appendix A contains a use case diagram and brief textual descriptions of each case.

3.2.3 Class design

With both interface and functionality largely decided upon, I could outline the class structure of PeerView. The resultant class diagram is shown in Appendix A, and the following describes the rationale behind the design.

The design was kept as simple as possible to avoid complications and allow easy modifications at later stages. The classes that make up the design are divided into interface, entity, and control classes. The *interface classes* are those implementing the GUI, the *entity classes* are those that contain and manage the application data structures without performing any but minor operations on them, and *control classes* are those that mediate between model and interface by modifying the former and updating the latter in response to user interaction. This division is a very simple object pattern, but is nevertheless not adhered to with absolute rigour as flexibility of design had priority at all times. For example, the `Constants` hierarchy in Diagram A.2 is an appendage to the model classes in that it does not represent a data structure, but a repository of constants for use by one or more other classes. This allows constants to stored centrally, accessed through get/set method pairs, and appear where used in the source code as meaningfully phrased function calls rather than string literals. For more

[3] Movement along the x or y axis.

on this, see the source code in Appendix C or Section 3.3. Also, most of the control code is placed in central control classes, namely `ClientManager` and `ServerManager`, and almost all the entity data in central entity classes, namely `ClientData` and `ServerData`. This was done because the interface design clearly indicated that most data structures would be either read from or written to by more than one interface class via the appropriate control class. There was therefore little need to complicate the design by having multiple entity/control pairs (one for each interface class in the extreme case).

The many classes that implement *listener* interfaces are event handlers. These were added as implementation progressed rather than at design time, for although event handling is the standard notification mechanism in Java and the need for handlers therefore was obvious, the exact number and interfaces were difficult to determine at this stage. An attempt to incorporate handlers in the early design was therefore apt to be inadequate. The decision to add event handlers only during implementation seemed unproblematic since handlers are highly standardized, and can be nested in other classes, so I deemed the risk of this causing unplanned structural design change to be minimal.

3.3 Implementation

Implementation was done over a 6 month period from the middle of May 2000 to November 2000 using the VisualAge for Java 3.0 IDE (Integrated Development Environment) [58]. As for implementation methodology, I decided to proceed iteratively and by small increments, i.e. frequent builds[4] and application tests. This may seem an obvious approach, given that it is probably how most small scale projects are carried out, but there are alternatives. In [42], Mills et al. discusses *cleanroom* development where developers are forbidden to test and execute their code while writing it, and tests are based on statistical analysis. The method is directed towards projects with no or low fault tolerance, i.e. projects where the code produced must have a low error to KLOC (Kilo-Line-Of-Code) ratio, and is intended for use by teams of developers as it relies heavily on peer review. A different approach that seems more attuned to development of applications like PeerView is *Extreme Programming* [6] which supports the notion of small increments. It also advocates programming in pairs as a method of ensuring continuous peer review, so I could obviously not adopt extreme programming in toto. Instead, I considered its recommendations as validation of my personal experience from earlier projects which was that small increments and constant feedback stimulates motivation and helps reduce the error rate.

The order in which design features were added to the framework produced during the design phase[5] was determined by how essential they were to the end user, or, to put it differently, how indispensable they were to the application. For example, it would be possible for an end user to make sensible use of PeerView if it provided nothing more than a client application with a scalable desktop and functionality for adding artifacts to it. It would not be possible if that user had only a working server application with a fully functional communications system but nothing else. This approach was chosen because it ensured that a usable, if amputated, system would always be in place for demonstration and preliminary releases if the need arose, and because it allowed realistic interface and functionality testing to take

[4] A build is the compiling and linking of all components making up an application.

[5] The interface classes created during GUI design and the skeleton code for entity and control classes.

place during development. The major features are discussed below in the order they were implemented. A major feature in this context is one whose implementation is non-trivial. An example of the opposite, a trivial feature, could be transfer of string literals from one list box to another, or reading the contents of a set of files.

3.3.1 The scalable desktop

The scalable desktop is implemented by the class `ClientPanorama`. The term "panorama" is used throughout the PeerView implementation and documentation rather than the phrase "scalable desktop" used during design, because I reasoned that the idea of a panorama window providing overview of a vista had more intuitive appeal to the average user than the more technical notion of a scalable desktop. `ClientPanorama` is derived from the Jazz library class `ZCanvas` which implements a simple scenegraph structure, a drawing surface and a viewport camera (shown in Figure 3.2).

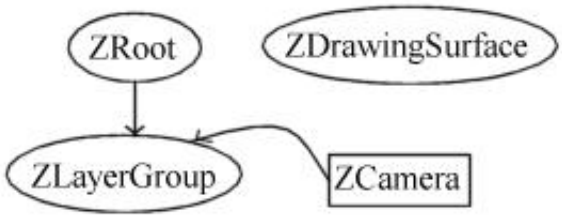


Figure 3.2: The `ZCanvas` scenegraph.

A scenegraph is a collection of hierarchically organized nodes that specify the structure of the scene that is rendered on a users's display. The nodes can be either *leaf nodes* which have no descendants or inner nodes, so called *group nodes*, from which other group nodes or leaf nodes can descend. Operations applied to a group node affects not only that, but also its descendants which allows operations to be applied in a flexible manner to arbitrarily sized subsets of the scenegraph, including the entire scenegraph if operations are applied to the unique root node. Group nodes can be categorized by the types of operations they support and Jazz includes such group types as `ZTransformGroup`, `ZStickyGroup` and `ZInvisibleGroup` which can be used to translate, scale and rotate their descendants, to make them appear at a fixed magnification and to be rendered invisible in the scene, respectively. A separate type of node is the *camera node* which can be attached to the scenegraph at an arbitrary position and used to render an image of the corresponding scene at that position. This means that one can place multiple cameras in a scenegraph and have different scene renderings produced without making changes to the graph itself.

The `ZCanvas` scenegraph contains the minimum of objects required to render a panorama, namely a `ZRoot` object with one child, a `ZLayerGroup` object which groups the set of nodes descended from it. Jazz has several types of such group nodes, most of which allows one to apply an operation to all nodes descended from them by applying that operation to the group node. The `ZCamera` node represents a viewport onto the nodes in the scenegraph and the `ZDrawingSurface` the surface onto which the camera view is projected. Artifacts are added to a `ClientPanorama` object by inserting a `ZVisualLeaf` node under the `ZLayerGroup` of the `ClientPanorama` (or rather, its `ZCanvas` super class). A `ZVisualLeaf` is a leaf node in the scenegraph which has a visible component that can be rendered in the scenegraph's `ZDrawingSurface`. An artifact is translated by adding a `ZTransformGroup` node above the

`ZVisualLeaf` corresponding to that artifact and then applying a translation operation to that `ZTransformGroup`. Before adding more features to the panorama class, I implemented the basic functionality of the `AddFiles` and `RemoveFiles` classes so that artifacts could be placed on and removed from the panorama. This was in keeping with the overall aim of adding functionality in the order of usefulness, but mostly a practical concern since it would be difficult to test operations on the panorama if there was no way of inserting objects into it.

Scaling

The entire panorama can be scaled, but not individual artifacts. The reason for not allowing scaling of a single artifact is not only the overarching decision to maintain simplicity of design in the initial version of PeerView, but also that such a feature could easily be counterproductive. If individual artifacts could be scaled, the resulting panorama would be a mosaic of objects at different positions along the z-axis. Translating an object would then become a potentially user-unfriendly operation since a translation by Δ of an object o at z-coordinate α along either the x or y axis is transformed to a translation by $\frac{\Delta}{\alpha}$ in the `ZDrawingSurface` rendering that is visible to the user. The user would then experience differing responses to identical inputs as an indication to translate an object o at z-coordinate α by Δ would appear to move o further than if another object o' at z-coordinate $n * \alpha$, where $n > 1$, was translated by the same amount.

The scaling from magnification Z to the target magnification Z' is done by interpolating between the two values over a period of length l, where l is user adjustable using the preferences Section of the PeerView client application. The interpolation follows a slow-in-slow-out trajectory and the display is updated after each step, so to the user a scaling operation appears as a smooth animation from Z to Z' that begins and ends with a gradual slowdown rather than an abrupt halt.

Performance

When implementing the translation and scaling operations, it became clear that poor runtime performance would be very detrimental to the usability of PeerView. That is, if those core operations could not be executed smoothly and with little delay on the average system, then users were likely to be exasperated which is obviously not conducive to end user satisfaction. The default behaviour of the Jazz renderer is to render objects on the drawing surface at one of three levels of detail: low, medium or high. These settings of course reflect a trade-off between quality and speed of rendering. I was aware of this at the outset, i.e. when beginning implementation and so had expected that quality could simply be reduced during critical operations to increase the speed at which they were carried out. However, this was not sufficient, for although performance was satisfactory on my system which must today be considered an average configuration PC, it was too sluggish on other systems (such as thin/dumb terminals) that for example did not have dedicated terminal hardware for graphics display, but instead received images from a server through a local network. I therefore modified the renderer so that objects were rendered as filled grey rectangles when translation and scaling operations were carried out. This improved runtime performance significantly, but not enough for all systems so there are still minimum system requirements when using PeerView. However, given the hardware of the average user today, I do not expect this to be a serious problem.

Layout

The panorama should support automatic layout. This was decided during design since it is a feature known from other desktop GUIs that can help the user restore order in a cluttered display, cf. for example the "Arrange" item on the desktop pop-up menu of the MS Windows platform. Layout is performed by a *layout manager* which is a class that accepts a set of nodes and then arranges them using a layout algorithm that may be arbitrarily sophisticated. Currently, PeerView 1.0 provides only the `GridPanoramaLayoutManager` class which is derived from the abstract class `ClientPanoramaLayoutManager` that forms the root of the so far small layout manager class hierarchy. A `GridPanoramaLayoutManager` arranges a set of artifacts in a square grid pattern where spacing is determined by the largest of those artifacts[6]. The user may specify the type of layout manager to be used from the preferences dialog of the client application, but as mentioned, only grid layout is currently available. The layout manager is called when the user inserts artifacts into the panorama and when she selects the "Arrange" item from the panorama pop-up menu. I chose the grid layout algorithm because it was simple to implement and the default layout in most other desktop GUIs. There is no inherent limitation to the level of complexity of the layout manager, though, and one could envision managers that laid out according to semantic criteria such as content, type (text, graphics, etc.) and size in addition of course to the wide variety of geometric layouts that can be conceived. So far, however, I have not found the time to implement any of these, but since a layout manager is simply a class derived from `ClientPanoramaLayoutManager` that adheres to a particular interface and accepts a set of nodes, it should be possible for other developers to add such managers to PeerView 1.0, if desired.

Additional features

After experimenting with the panorama, I found out that it would be useful to be able to resize artifacts and to fix their position, i.e. "pin" them to the panorama surface, so to speak. This would give the user an acceptable degree of freedom in customizing the display without increasing development time significantly. I therefore introduced a resize feature to the `ClientPanorama` class and a corresponding item to the artifact pop-up menu. The operation is carried out by dragging a corner of the artifact in question to the desired position and then left-clicking the mouse. The artifact retains its size until removed or resized again. The ability to fix an artifact at a given position was implemented by maintaining a lock for each artifact (implemented as a boolean data member) and then excluding locked artifacts from any layout operations. The lock can be toggled from the artifact pop-up menu which also allows the user to toggle the default lock setting for any new documents added to the panorama. Both the resize and the lock feature necessitated changes to some of the method definitions in the `GridPanoramaLayoutManager` class.

3.3.2 Message and progress bar

At the very bottom of the client application main window there is a small rectangular area which is used for displaying error and status messages and for indicating progress for operations of some duration. This area, called the message area or message bar, can be double-clicked to

[6]This method of spacing was introduced after implementing the resize feature, see Section 3.3.1

expand into a window containing a log of previously displayed messages. Implementation of the message area was mostly straightforward, but one aspect was non-trivial, namely how to make the progress bar update and display and at the same time perform the operation whose progress the bar was indicating. I tried first to have the code implementing the operation force the progress bar to update at each step of execution, but I learned from a newsgroup conversation that this approach reportedly was grossly inefficient. I decided to rely on this information and implemented the message area as recommended, namely by executing the code for a task whose progress was to be indicated in a separate thread. That code will then modify the progress bar which runs in the event-handling thread[7] and is therefore automatically updated by the system when that thread is executed.

Because of Java's platform independence, the issue of multi-threading is made somewhat more complicated than it appears from the above. Different operating systems implement different multi-tasking policies which affects the way in which threads are allotted processor resources. Notably, the UNIX and Solaris operating systems use *preemptive scheduling* where executing threads are only taken out of their running state when they *yield* control or are forced out because a thread with higher priority requests execution. The Windows NT operating system, however, uses *time slicing* as its multi-tasking control mechanism where processes such as threads are given a fixed size time-slice in which to execute. When that time-slice expires, the thread is put out of its running state and the next contender put into its running state instead, regardless of whether the first thread yielded control or not. In practice, this means that on systems that use preemptive scheduling, the threads started by PeerView may be preempted, i.e. put out of their running state, by unrelated processes that happen to have higher priority and be executing on the system in question at the same time. This will sometimes produce noticeable delays in updating the PeerView interface which the user will most likely experience as erratic changes in the length of the progress bar. The code for launching tasks in separate threads is in the code listing for the `ClientApplication` class in section C.2 (e.g. the inner class `RemoveDistributedDocumentsFromPanorama`).

3.3.3 Communications layer

Before the group management and discussion forum functionality could be implemented, the basic functionality for message and data exchange had to be in place. It is perhaps debatable whether this was in accordance with the decision to add features in the order of usefulness to the end user, but without means of transmitting group and message data it would be difficult to test the components responsible for those types of data. To create a communications layer, I had to add some functionality to the server application skeleton code produced during design. Specifically, the server is responsible for maintaining what the JSDT documentation refers to as a *registry*, i.e. a small database listing the clients and sessions registered with that server. The server also maintains a concurrently updated database of group information, messages and discussion fora, but most of that functionality was added later when both the group directory and the discussion forum components which generate that data had been implemented. Initially, only the methods for creating and destroying a registry and for opening channels of communication was implemented on the server side. On the client side, code was written for methods in the `ClientManager` class to enable it to connect to the server, to join

[7] Java Swing events are always handled by the thread which started the application generating those events.

a session (which, cf. Section 3.2.2, corresponds to a group in PeerView group), and to send and receive data through communications channels.

3.3.4 Group directory

The group directory is shown in Figure 2.2. It provides a listing of available groups and functionality for joining a group, adding or editing one or deleting one from the listing. The interface had been constructed during the design phase so what remained was implementing the appropriate functionality and associating it with the push buttons at the bottom of the group directory dialog box. The add, edit and delete operations were "database primitives" in the sense of being the standard operations for simple interactive data structures which typically provide the ability to insert and delete elements as well as replace existing elements with new versions. Implementing these as operations on the local data structure, i.e. the one whose contents are displayed in the listing above the panel of buttons was therefore straightforward. However, after each operation, that local data structure is submitted to the server which maintains the master copy of the group listing. The server then updates the master to reflect the changes in the local structure and subsequently resubmits the master to all client currently connected to the server. The data structure is typically quite small so transmitting it from client to server in its entirety does not add significant overhead compared to just transmitting a notification of the operation performed and is simpler to implement and extend (new operations can be added without having to introduce a new type of notification, for example).

The "Join group" operation is more composite than the data management operations discussed above. To join a group, the following must be executed, assuming that a group was actually selected when the user opted to join:

1. If the user is currently member of another group, then:
   (a) remove (the copies of) all other group members' artifacts from the user's panorama;
   (b) notify the server that the user has left so that the master copy of the group directory can be updated to reflect the change in number of group members and total data volume;
   (c) notify the other group members that the user has left so that they can remove (their copies of) the user's artifacts from their panoramas.
2. Join the selected group and then
   (a) broadcast copies of all the user's artifacts to all other group members;
   (b) request copies of all other group members' artifacts and add them to the user's panorama when received.

Minor details have been omitted from this definition; they appear from the source code in Appendix C. The operation was implemented so that the client leaving a group need only remove its copies of other group members' artifacts and then actually leave the JSDT session corresponding to the group. The server will then be automatically notified since it monitors all channels of all sessions for leave and join operations. Upon notification, the server will

notify all other group members that the client has left the group and they will then update their panoramas accordingly. The server also changes the master copy of the group directory listing and broadcasts the updated copy to all active clients.

The group listing master copy maintained by the server is written to disk when the server exits and read again when it is next started.

3.3.5 Discussion forum

The discussion forum has two components: the *topic tree* which is a collapsible tree structure that organizes hierarchically a set of nested topics, and a message pane (also referred to as the message area when addressing the user since "pane" might not make intuitive sense), where the messages selected from the topic tree are displayed. Each time a user selects an item from the tree that corresponds to a message added by herself or another user, the client application requests that message from the server which maintains a database of all discussions and messages. The server responds to the request by sending a `Message` object which is a simple data structure containing the message text and relevant meta-data such as author name, date and time of composition, etc.

The server database of messages and discussions is written to disk when the server exits and read again when it is next started. The storage format is plain text, which implies that compression might reduce the size significantly. I found it unlikely, however, that the size would become unmanageable great in realistic use scenarios if the data was stored in its uncompressed form and that therefore became the storage format for all such plain text files maintained by the server.

3.3.6 Artifact updating

To provide an accurate representation of a set of artifacts, the PeerView client must ensure that the objects placed in the panorama are updated at regular intervals. The decision to make the exact interval user adjustable was made during design as this affected the interface, but the specific method of updating was left undecided since I reasoned that choosing the best method would require some experimentation which required a code scaffolding, i.e. there had to be a panorama and artifacts for experiments on them to take place.

A naïve and failsafe method of updating artifacts can be formulated using pseudo-code:

```
1.  for each artifact α
1.1.  if ( ( changed( α ) == true ) AND ( group( owner( α ) ) != null ) )
1.1.1.  broadcast( α , group( owner( α ) ) )
2.  goto 1
```

The functions called in the above are assumed to be defined elsewhere. Function `changed( α )` returns `true` if the artifact α has been changed since the function was last called with α as argument. Function `owner( α )` returns a unique identification of the owner of the artifact α, and function `group( id )` returns a reference to the group to which the user corresponding to `id` belongs, or `null` if he or she does not belong to a group. The function `broadcast( α, g )` broadcasts a copy of the artifact α to every member of the group except `owner( α ). g`.

Since PeerView 1.0 supports both text and bitmap artifacts, this is a potentially very inefficient and costly method of updating as the size of a typical bitmap image at resolution

1024x768x24 may be in excess of 3Mb, depending on the storage format. An alternative is to transmit only the difference between an artifact before and after a modification, as outlined below:

```
1.  for each artifact α
1.1.  if ( ( changed( α ) == true ) AND ( group( owner( α ) ) != null ) )
1.1.1.  broadcast( diff( α ) , group( owner( α ) ) )
2.  goto 1
```

The function `diff` computes and returns the difference between the artifact α before and after the most recent change was made. Each client that received the difference data d would then execute the following:

```
1.  α = combine( α, diff( α ) )
```

The function `combine` computes and returns the artifact α' that results from merging the artifact α with the difference returned by `diff`.

The runtime performance of this method can be enhanced by compressing the difference packages sent between clients, and it can be further improved by adjusting the quality (and thus time complexity) of compression to match the processing power to network bandwidth ratio of the system it is executed on. Such adjustment can be either automatic or user definable, depending on the implementation effort one is willing to invest.

Simple back-of-the-envelope calculations show that an implementation of the difference algorithm using for example the DEFLATE algorithm [16] for compression will outperform the naïve algorithm by an order of magnitude or more:

Notation:

d the average size of a document in bytes

s the average size in bytes of the difference between a document before and after a modification.

u the number of update operations pr. sec.

c the compression factor.

N the naïve algorithm bandwidth consumption pr. sec.

C the compressed algorithm bandwidth consumption pr. sec.

Assume:

$d = n$
$s = \frac{n}{10}$
$c = 4$
$u = 3$

Then:

$$N = d * u = n * 3 = 3n$$

$$C = \frac{s * u}{c} = \frac{\frac{n}{10} * 3}{4} = 0.075n$$

and if for example $n = 10\text{KB} = 10 * 1024$ bytes $= 10.240$ bytes then $N = 30.720$ bytes and $C = 768$ bytes.

Although these were loosely based calculations, they convinced me that experiments were unnecessary: the difference algorithm was the best choice, since its time/space tradeoff was best by a wide margin.

However, my first attempt at implementing a difference algorithm was hurried and not well enough thought through as a discussion with Jyrki Katajainen made clear. I had gone ahead with my initial idea for a difference algorithm which consisted in computing the difference between two byte arrays representing a document before and after a change by performing a logical XOR operation between the overlapping portions and appending whatever remained (which I surmised was likely to be neglible) to the difference which was also stored in an array. This difference array would then consist of long strings of 0's and 1's indicating areas of change and areas that were unaffected by the modifications made since the previous update. I then used the Java `Deflater` class [50] to compress the difference array before it was broadcast to other clients who in turn used the Java `Inflater` class [50] to decompress the difference array. They then combined the difference array with their old version of the document it was associated with by performing the same XOR operation as was used to derive the difference. Because of the typical contents of the difference array (long sequences of 0's and 1's), the compression ratios were quite impressive and the entire arrangement seemed to simple and symmetric that I conveniently forgot to think actual usage scenarios through. The truth of the matter is that this XOR difference approach is only effective if the size of the array storing the document before the change is exactly the same as the size of the array storing the document after the change. Only then will the difference array contain only the difference between the two input arrays, punctuated by long sequences of identical symbols and thus yielding the favourable compression ratios compared to transmitting the entire document [8]I was after. If, on the other hand, the change to a document resulted in a change to its size then the difference array would contain a sequence of identical symbols of length equal to the distance from the beginning of the document to the beginning of the changed portion. The remainder of the difference array, i.e. the portion corresponding to the beginning of the change in the document to the end of the document, would be an XOR combination of the document before and after it was changed, and because of its change in size, those two versions would be shifted relative to each other. If we assume that, on the average, a document will be changed at position $\frac{n}{2}$ where n is the length of the document (and we ignore the size of the change itself), the difference array can only be compressed to about half the size that the original document could since it is, on average, half as "random", to put it informally. An example might make this clearer:

[8]Typical compression schemes will reduce a long sequence of identical symbols to a representation consisting of the symbol being repeated throughout the sequence and the number of times it is repeated, i.e. the sequence length.

	Modification without change of size	Modification with change of size
Original array	000110011001111110000111111	000110011001111110000111111
Modified array	000110011001100011000011111	0001100110001
Difference array	000000000000011100000000000	000000000001011111000011111

As the above suggests, the XOR difference array algorithm was far from optimal in that it only works satisfactorily if a change to a document does not result in a change to its size which is, of course, an overly restrictive limitation. It is, however, the algorithm used in the current implementation of PeerView as I have not had time to implement a better alternative. Some such alternatives are described in [31], where Hunt et al. survey different *delta algorithms* as algorithms for computing the difference between successive versions of a document are typically called in available research literature. They find that the traditional `diff` delta compressor found on the UNIX operating system is outperformed by more recent algorithms based on Ziv-Lempel techniques (such as the DEFLATE algorithm used in PeerView) and they present an alternative that they call *vdelta* which performs better, both in terms of compression ratios and in terms of compression/decompression times, than `diff`, `diff` combined with `gzip` compression and `bdiff`, a successor to `diff`.

The vdelta algorithm therefore looks like a promising candidate for use in PeerView. Indeed, the authors suggest that the algorithm is well suited for use in such applications [31, p. 10]. I have, however, not given implementation of an improved algorithm top priority since the current version functions satisfactorily: typical compression ratios are better than 1:15 (averaged over text and graphics) so on all but very large documents that are changed with short intervals, it performs quite well. If the system was to be developed further, however, the algorithm would have to be replaced since it consumes an unneccesary amount of bandwidth as the number of users increases.

Chapter 4

Experimental evaluation of PeerView

4.1 Introduction

Evaluating groupware is difficult because it combines the traditional problems involved in evaluating single user system with the complications introduced by scaling to multiple users [3, 45]. Among those complications are both practical concerns, such as recruiting and compensating users, planning and conducting experiments and scheduling them as well as methodological concerns such as how to best gather and use experimental data. This makes it difficult to formulate a suitable procedure for evaluating groupware. In fact, Baker et al. [3] claim that no cost-effective techniques to that end exist and that one is therefore consigned to adapting existing, single-user evaluation techniques to a groupware context.

I opted for a controlled experiment with a small group of participants who were asked to carry out a set of tasks using PeerView and subsequently to fill out a questionnaire and comment on their experience using the software. The reason for making this choice is discussed in further details in Section 4.3. The experiment itself was recorded on audio tape[1] and combined with the written feedback from the three participants, it yielded substantial information on the usability of PeerView.

The below discussion is based on the materials used in the course of the experiment, all of which can be found in Appendix B.

4.2 The experiment

The experimental evaluation of PeerView was carried out on March 14th, 2001. The setting was a room in a Copenhagen computer café equipped with four medium-range PC systems running Microsoft Windows 98. The machines were fitted with 17" displays and Pentium II processors and were connected by a local area network. On one of the machines, the PeerView server software had been installed in advance since I did not consider the installation and configuration relevant to this end user experiment whose primary focus was to be on the

[1] Circumstances did not allow for video recording, but as appears from the transscript (a written reproduction of the dialogue on the audio tape) in Appendix B.2, the participants' comments and questions were quite interesting in themselves and gave enough information to deduce many of their actions.

client side software. The other terminals contained only the software installed by the café proprietors, i.e. mostly communications software and computer games.

The three participants were a 27 year old woman (referred to as "Rikke" in the transscript) and two men aged 26 ("Thomas") and 28 ("Paul"), respectively. They all had good working knowledge of how to use computers on a day-to-day basis, but had no super-user skills or prior experience with either groupware or zoomable interfaces. They were each assigned a PC and handed a task list (reproduced in Appendix B.2) which they were asked to read and proceed with. The tasks where linearly structured with each being dependent on its predecessor on the list, but required use of almost all the functionality in PeerView and therefore constituted a thorough walkthrough of the system. This setup was chosen because it allowed me to focus on the participants' behaviour and performance rather than on explaining the intent behind a set of open tasks with no specification of how to perform them.

The first task on the list was to download and install the latest version of PeerView from its website [54]. At the time of the experiment, the latest version was 0.95, i.e. still a beta but close enough to a stable release (1.0) that I found it acceptable to conduct an experiment since I did not intend to add any major functionality nor expect to find any critical bugs[2]. The download process took approximately 10 minutes which was considerably longer than expected. The delay was caused by severe network lag and thus probably an accurate reflection of the fluctuations that users would experience under less controlled circumstances. Upon successful download and installation which took place without much difficulty, the participants began carrying out their tasks. One of the participants, the 28 year old man, had been given a task list which was slightly different from that given to the two others. It instructed him to create a group for the other two to join and to edit a shared document to determine whether changes were distributed once they had joined. Both he and the other two participants were later asked (by the task list) to contribute to the topic tree corresponding to the shared document he had added. The difference in task lists was necessary to strengthen the collaborative aspect of the experiment. By setting up a group and have all users join it, and then collaborate, albeit in a rudimentary fashion, on a shared document, the experiment came to encompass not only the single user dimensions of PeerView, but also its collaborative functionality.

The first task for all three, however, was to adjust the preferences so that their name and the correct server information was entered. They did this by modifying the settings in the preferences dialog box and then restarted the entire program to make the settings take effect. The last step was necessary because instant application[3] had not yet been implemented. Afterwards, the participant assigned the task of creating a group carried that out while the other two added a number of documents to their respective panoramas. They then moved on to carry out the other tasks on their task lists as appears from the material in Appendix B.

4.3 Discussion

The PeerView usability experiment was limited in scope by having only three participants and a duration of only two hours including the time taken to fill out the questionnaires. In spite of

[2]Program errors capable of triggering runtime failures such as system crashes.

[3]Often signaled by an "Apply" button in such dialog boxes.

this, it yielded a fair amount of useful information and gave me reason to believe that increasing the duration and number of participants would probably have given a quantitatively, not a qualitatively different amount of information. I therefore trust the results to be substantial enough to safely draw conclusions about the usability of PeerView in its current form. The use of several techniques — controlled experiment, user observation, questionnaire and informal discussion after the experiment — was chosen to lend the findings more credibility. That particular set of techniques was decided upon because I deemed it well suited for gathering a diverse set of data and because it comprises the most widely used forms of evaluation among groupware researchers. In [45], Pinelle analyses the evaluation performed in 32 of 45 articles on groupware (the remainder did not contain formal evaluation), and finds that laboratory experiment is the most commonly used evaluation type, and that user observation, interview and questionnaire techniques, in that order of prevalence, are the most popular ways of carrying out the evaluation.

The experiment produced information on both the *single user* and *collaborative* usability of PeerView, as appears from the transscript (Appendix B.2). After the user[4] assigned with the task of creating a group had done so, the other users joined the group and subsequently received copies of the artifacts that other users had placed in their panoramas. They were then each given the task of contributing to the topic tree associated with a specific document (referred to as "test_ gruppe.txt" in the transscript) added by the user who created the group (the only document he was asked to put in his panorama). Towards the end of the experiment, the same user modified "test_ gruppe.txt" and the other users checked their copies to see whether the change had been distributed correctly, i.e. whether their copies of that particular document had been updated to reflect the change. I had, of course, tested the update functionality separately, so the purpose was not so much to see whether it worked according to specifications, but rather to round off the collaborative component of the experiment.

4.3.1 Experimental findings

The experiment participants indicated or remarked repeatedly (e.g. transscript item 28,40,62 and questionnaires) that they had problems understanding the system dialogue or the consequences of the actions they performed. This was mostly due to the dialogue being in English, using unknown or counterintuitive terminology (such as referring to the nested topic structure in the bottom portion of the window as a "topic tree") or to the interface being less informative than could be desired (as in the case where one participant asked how he could tell whether he had joined a group [transscript, item 46]). The participants did not use the help system although they were encouraged to do so which was probably due to me being present while the experiment took place and answering questions on how to interpret system messages and other trivial issues such as which keys to press to produce a tilde character).

There were a few concrete interface problems. One was that highlighted text appeared at a different scale from the highlighting itself [transscript, item 67], which made it seem as if a smaller portion of text was highlighted than was actually the case. This confusing phenomenon was referred to as an optical illusion by the participant who noticed it, and his subsequent attempts to align highlighting and text by scaling the document to 1:1 magnification further revealed that panorama scaling was perhaps overly sensitive to user input since he

[4] He is referred to as "Paul" in the transscript.

had difficulty controlling it. An unrelated problem was discovered by the woman participant who experienced difficulty with the visible documents box in the right hand portion of the toolbar panel. Her monitor operated at a lower resolution than the other monitors and as a consequence the visible documents drop-down box could not expand downwards. In effect, it "dropped up" rather than down by expanding in the upwards direction. This rendered its contents difficult to access as indicated in [transscript, item 42]. None of the participants seemed to notice or have use for the draggable resize dividers that separate the topic tree from the message area in the bottom of the display and the panorama from the discussion forum. At one point [transscript, item 54], I tried to draw the participants' attention to the dividers to see whether they ignored them because they were unaware of the resize functionality or because they had no need for it. None of them used the dividers after I had made them aware of their purpose, but I doubt that can be taken as indication that they are unneccessary since different participants seemed to grapple a bit with fitting a document in the visible viewport or seeing the title of a topic in the topic tree. They might therefore have benefited from the resize functionality if they had been more comfortable with it. It would probably be desirable to somehow communicate more clearly that the dividers bars can be used for resizing, perhaps by annotating them with a label containing descriptive text.

A more abstract, general problem reported mainly in the comments section of the questionnaires was that the participants had difficulty understanding the context of their actions because it was unclear what audience PeerView was aimed at. This became clearer during the informal discussion afterwards when I explained the idea of making a prototype design before making a fully fledged program, but the participants nevertheless indicated that they felt somewhat bewildered during their work with PeerView. I suspect that this reaction is typical when users meet a program with a set of expectations shaped by work with commercial applications aimed at solving specific work problems. If development had gone through several iterations, each concluded by experimental evaluation, the problem could have been countered by using the same group of test participants each time since they would then be accustomed to the work process and its premises. The drawback of that approach is of course that such a group of seasoned test participants would quickly become unrepresentative of the user audience at large, thus probably degrading the value of any experimental findings derived from their activities.

The collaborative aspect of the experiment (as described in Section 4.2) was too limited for me to draw any firm conclusions, but it did seem to indicate that the participants had little awareness of each other. This applies to both group awareness (in the sense of being cognizant of the other participants) or workspace awareness (being aware of the others' actions in the workspace) as indicated by [transscript, item 48, 74] where users display signs of not feeling part of a collaborative unit, seemingly in part because the PeerView interface does not make it sufficiently clear that they are. When addressing me, the participants occasionally made references to what they had overheard me saying to other participants which could be taken as suggesting that they would benefit from more direct communication facilities such as the "chat" that was planned for, but unfortunately not included in the PeerView 1.0 release due to time restrictions.

4.4 User evaluation

After the experiment was concluded, the participants and myself sat down to fill out the questionnaires handed to them and to discuss the experiment. The questionnaires were worded in a way so that they could be understood without specialized knowledge to minimize the potential for misunderstandings. They were divided into three Sections:

- A set of Likert scale propositions regarding specific usability issues. A Likert scale can be used to have informants indicate on a scale ranging from complete agreement to complete disagreement the degree to which they agree with a proposition.
- A semantic differential schema where the participants where asked to indicate on a bipolar scale the degree to which they felt a set of descriptive phrases applied to PeerView.
- A free form comments Section where participants where asked to write any comments or remarks they might have about PeerView.

The overall assessment of PeerView was favourable, but the comments indicated some of the usability problems discussed earlier, namely that the terminology used by the PeerView interface in its messages and widgets was at times difficult to understand, not only because of the language being foreign to the participants who were all Danish, but also because they used terms that were unknown to them. Another problem was that participants lacked a clear statement of whom PeerView catered to. One participant wrote that: "I have the impression that you would quickly grow accustomed to the programme and grow to like it in the proper setting" [questionnaires]. But the subsequent discussion revealed that, like the others, he had difficulty pinpointing exactly what that setting should be. I suggested that additional functionality, such as that to be included in InSiter, might make it more useful to them and their work colleagues as well as other groups, and they seemed to agree that enhanced awareness in the form of video mosaics of coworkers (for example in the style of PortHoles [17]), chat facilities and different workspace contents at different magnifications (semantic zooming) were worth introducing.

Chapter 5

Implications of PeerView 1.0 for future systems

5.1 Introduction

The PeerView 1.0 prototype is exactly that: a prototype for new ideas whose purpose is to demonstrate their feasibility and viability. It is intended as both a test vehicle, a useful tool and a learning experience on which to base design of future systems. In this Section, I will analyse *a posteriori* the design, implementation and development of PeerView 1.0 to examine how a future rendering and review system (outlined in Chapter 6) can benefit from the findings of the PeerView project.

Of course, the below applies not only to successor systems, but also to PeerView 1.0 itself. The findings and conclusions might therefore conceivably be incorporated into the current version of PeerView to yield a significantly upgraded product which could aptly be called PeerView 2.0! However, I did not intend PeerView as an "open-ended" project and the experience from building version 1.0 is so comprehensive that I have found it most appropriate to use a different name, namely *InSiter*, for the successor project. PeerView development may continue, though, since the project has been made open source. Time will tell.

5.2 Design analysis

5.2.1 Introduction

The discussion in the following Sections is mostly general, concerned with the guidelines for design that can be derived from PeerView. Details are only brought in where appropriate to this discussion as the purpose is to facilitate future development, not to formulate specific improvements to the current PeerView design.

5.2.2 Interface

In retrospect, it is clear that the design of the PeerView GUI rested on implicit assumptions about the suitability of metaphors as design devices. That is, by readily adopting the desktop metaphor (for the scalable desktop) and a debating society metaphor (for the discussion forum)[1], I accepted the commonly held belief among GUI designers that design based on metaphors equals user friendly design [9, p. 5]. This assumption is not necessarily justified, as Blackwell documents in his Ph.D thesis. Based on comprehensive experiments where novice and expert users were asked to solve various exercises using metaphorical and non-metaphorical diagrams and notations, he concludes that [9, p. 164]:

> "Most of the experiments described in this thesis ... have tested the assumption that diagrams are universally beneficial in problem solving and design. This is obviously not the case — even within the scope of these studies, a simple distinction between 'experts' and 'novices' has revealed not universal benefits, but a large difference between individuals with different amounts of experience."

How, then, can one account for the inarguable success of the desktop metaphor as popularized by the Apple Macintosh and Microsoft Windows operating systems? Blackwell does not attribute this success to the metaphor as such, but argues instead that [9, p. 160]:

> "The demonstrable advantages of graphical user interfaces, as with many types of diagram, can be explained in other terms. Direct manipulation — representing abstract computational entities as graphical objects that have a constant location on the screen until the user chooses to move them — facilitates reasoning about the interface by reducing the number of possible consequences of an action ..."

This seems to validate the choice of a desktop GUI design, although not by virtue of its metaphor design philosophy. I believe a similar argument applies to the discussion forum design as it relies on a form of direct manipulation that is both familiar to many users (for example from the graphical directory browsers found in many operating systems) and fits the description given by Blackwell. This seems to be corroborated by the predominantly positive assessment made in the questionnaires that the evaluation experiments completed (see Appendix B).

A useful lesson to be derived from the above is that one cannot ensure a sound design by relying exclusively on textbook recommendations and established dogma in the design community. Instead, the interface decided upon during the initial design phase could be translated into a working prototype or a mock-up, i.e. a "scaffolding" with no real functionality, which could then be subjected to usability testing on a group of end users. The empirical data thus gathered could then be used to modify, if necessary, the interface design, and the test cycle should then continue until the GUI design is deemed satisfactory. Of course, the initial design would rely on a set of implicit assumptions exactly as the PeerView design did. This seems inevitable since end users cannot, in general, be expected to describe a priori and in detail their ideal

[1] I have no direct evidence to support that the original USENET design of a hierarchically organized newsgroup was inspired by debating societies or similar discursive fora, but my choice of that design was influenced by such considerations.

interface although most can, presumably, give constructive feedback on a concrete proposal presented to them. The designer should therefore ensure that his assumptions are grounded in empirical fact and well reasoned rationale, not only in commonly accepted practice and beliefs.

5.2.3 Functionality

It is clear that the actions that users can perform using PeerView constitute a minimum of functionality for a usable application. This is hardly surprising, considering that it is a prototype, but it does indicate that any successor system should not restrict itself to the ideas now explored if the scope of the research is to be increased. It should provide a fully fledged system that has the width and depth necessary to be useful in realistic work settings with large groups and large masses of data. The objective should not necessarily be to compete with similar, commercial systems, but to elaborate on the ideas embodied in PeerView, *and* demonstrate that they can "survive in the wild".

The bulk of user interaction in PeerView 1.0 is with the scalable desktop, i.e. the panorama on which artifacts are placed. The issues raised by adopting a desktop metaphor design were discussed in Section 5.2.2, and the rationale behind using a zoomable interface was presented in Section 3.1. The actions that users can perform on the panorama, i.e. its functionality, was limited to the essential operations as discussed in Section 3.1. Hence, there is ample room for improvement, both by extending the set of operations, although this would be merely a quantitative improvement, and by using a more sophisticated zoom mechanism which could make the entire interface qualitatively different. One such enhancement is *semantic zooming*, where the level of elevation (i.e. distance from the origo of the z axis) determines the representation used in the zoomable display. The reverse — that the choice of representation determines the elevation — has also been tried and is called *goal-directed zoom* in [59], where the authors describe a system that allows users to select an object and choose from a set of representations how it should be rendered. The system itself determines the appropriate form of representation at a given elevation by applying what the authors refer to as the Principle of Constant Information Density, which states that "the amount of visual information per display unit should remain constant as the user pans and zooms" [59, p. 306].

The Jazz library can be extended to support semantic zooming [8], so implementing such functionality is quite feasible using known technologies. I believe it could be of particular use in systems where the rendering was extended to arbitrarily large sets of artifacts and the process by which they were brought about. The reason is that it would allow a series of mutually supporting representations of group work to be rendered within the same display space and navigated in a convenient and familiar fashion. One example might be to have a project represented by a diagram at high elevations, by clusters of developers represented by photographs and brief biographical synopses at medium elevations and by grid arrangements of artifacts at low elevations, as done in PeerView. Chapter 6 elaborates further on this point.

Future rendering systems could of course abandon the idea of zoomable interfaces altogether and use entirely different modes of presentation. I believe, however, that ZUIs that support semantic zooming are a near ideal "$2\frac{1}{2}$D" alternative to the conventional 2D and 3D displays, both of which have inherent limitations, as discussed in Section 3.1. Consequently, future systems would benefit from extending the PeerView 1.0 interface both qualitatively,

by introducing elevation dependent representation through semantic zooming, and, possibly, quantitatively by giving users a wider selection of options than that listed in the PeerView 1.0 artifact pop-up menu.

In terms of support for collaboration, PeerView 1.0 gives users simple *workspace awareness*, i.e. who the other members are and what they are doing (to artifacts) in the shared workspace, and has very rudimentary or no support for other forms of awareness, such as *group-structural awareness* (awareness of people's roles in a group, their status and positions on issues) and *social awareness* (awareness of people's emotional and intellectual states) [47, p. 2]. In [26], Gutwin and Greenberg find a strong positive correlation between the quality of support for workspace awareness in groupware applications and the performance and satisfaction of participants in experiments based on those applications. Their subjects used a *radar view* window which is referred as a "miniature" [2] to get a global perspective on the workspace and to follow the activities of other users. PeerView 1.0 naturally supports the former of these activities, but not the latter and that issue should be addressed by any successor systems. In fact, Gutwin and Greenberg write that [26, p. 517]:

> " ...the main finding of the study is that adding workspace awareness information to the miniature — visual indications of viewport location, cursor movement, and object movement — can significantly improve speed, efficiency, and satisfaction. These awareness components should be included in shared-workspace applications."

This point seems to further underscore the qualities of a zoomable interface as compared to 2D and 3D alternatives which typically would require some dedicated miniature to be grafted onto the main interface to obtain the effect recommended by Gutwin and Greenberg. Zoomable interfaces allow an overview effect to be completely integrated in the ordinary operation of the interface, as PeerView demonstrates. Future systems could elaborate on that effect by integrating the features discussed in the above quote without subtracting from the interface's ease of use, i.e. without inconveniencing the user.

5.3 Implementation analysis

5.3.1 Introduction

In this Section, I discuss those aspects of the PeerView implementation that carry useful lessons for implementation of InSiter and of system implementation in general. As with design, those lessons are mostly general and not specific to implementation details, but the discussion does bring in concrete examples where necessary.

5.3.2 Implementation language and technologies

As discussed in Section 3.1, the implementation of PeerView was facilitated by using readily available component technologies to shorten development time. Jazz [34] and JSDT [?] were the most visible such component packages, but also the Swing interface classes that are usually

[2] A scaled down representation of the workspace proper.

thought of and used as an integral part of the Java Development Kit were a component technology that helped make implementation by a single developer possible. I experienced little difficulty using these component technologies and ascribe this in large part to the fact that all were written in pure Java, i.e. without use of code specific to a particular platform or architecture, so called native code. Also, Jazz, JSDT and Swing all have well designed APIs that make interfacing them with one's own code quite easy.

Judging from this experience, component-based development seems to be good practice but the following should be borne in mind:

- The interface of components should be well defined and well behaved, meaning that the API documentation should provide descriptions of both the syntax (how to use) and the semantics (what happens when used) of each component method along with a description of its side effects and complexity. If these are not available, the developer may realise that the savings in development time was too dearly bought with deficiencies in the end product. As Meyer notes in [41]: " ...the quality of a component-based application is defined by the quality of its worst component."

- The components libraries used should be compatible, preferably by being written in the implementation language of the application proper and by adhering to the same design principles. The problems that can be incurred by using code from separate languages are well known: one poignant example is the need to distinguish between different calling conventions for functions in `C++` code where for example one must specify that a function should be called with the `C` language convention by prefixing the function definition with `extern "C"`. The problems that may arise from differences in design may be more subtle, but no less real. One example could be to make use of two component libraries, one of which was highly optimized for some time critical operation such as real-time rendering, the other designed with security concerns as the primary concern and performance as a distant second or third. Such grinding incongruency between design goals could result in components cancelling out each other's respective benefits, making for a whole that is actually less than the sum of its parts.

- Component based development is not merely an question of picking off-the-shelf components, connecting them in some appropriate way and then executing the resultant mélange. The developer must negotiate between the ideal implementation of her design and the trade-offs involved in using component technology. In PeerView's case, for example, the Jazz library was used to implement the zoomable interface, i.e. the scalable desktop, but there were alternatives that had a differently balanced design. OpenGL [44] was one such alternative, as indicated in Section 3.1, which has a much broader range of graphics operations than Jazz, but no specialized suppport for the 2D zooming operations needed to scale text and bitmap graphics at interactive rates on even low-end systems [7]. One could envision a more sophisticated system that used a combination of zooming and 3D navigation (for example by means of a scale space representation [20, p. 110]), and might therefore have to rely on a library such as OpenGL, which in turn would mean that the minimum system requirements for end users would have to be raised significantly.

Component-based development therefore affects the entire development cycle, not just class

design and implementation. Trade-offs have to be made throughout to find the best balance between the envisioned end product and the degrees of freedom afforded by available component libraries.

5.3.3 Representation of properties and messages

An application like PeerView typically contains a set of mutable and constant values that define its behaviour and interface. An example of the former are error messages most of which are typically immutable and triggered by specific (exceptional) conditions in the application. An example of the latter are *preferences* which determine the general appearance and functionality of an application and can usually be defined by the user.

It is a basic convention in good programming style not to use so called "magic numbers" [40, find side], i.e. constants that appear by their literal value in the code and not a symbolic name. This applies not only to numerical constants, of course, but also to string literals such as user messages. Consequently, all such literals are represented by symbolic names in PeerView. The literals themselves are stored in designated classes called `ServerConstants` and `ClientConstants` and are accessed through access methods, not directly, so a literal whose symbolic name was `windowSize` would be accessed through a function call to a method named `getWindowSize`. This makes it possible to concatenate the constant with one or more arguments to the `get` method before the resultant value is returned to the user. One example is:

```
public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName,
String senderName)
{
return senderName + " has requested that " + documentName + " be removed from the
panorama.";
}
```

Since some constants are shared by the client and server application, the `ClientConstants` and `ServerConstants` are derived from the class `SharedConstants` which contains those shared constants and the associated access methods. Both the server and client constants class contain a set of default preferences which are constants used when the user has not specified any value for a specific property.

This design is not new; using classes of static constants accessed through methods is well-known practice in Java programming, but by applying this technique consistently as in PeerView, the textual component of the user interface is separated from the implementation. This means that if a different version must be installed, either to correct an error (a "bug fix") or to change the language, it can be done without requiring a new full install by the end user, or indeed a compilation by the developer. Users can instead download a small patch to the finished product and install that as instructed. This is a simpler method than the internationalization scheme described in [35] and could therefore be useful when the implementation should adhere to good style conventions, but not be bogged down in the details of internationalization.

5.3.4 Implementation of the distributed architecture

The distribution layer or data sharing mechanism of PeerView is implemented using the Java Shared Data Toolkit (JSDT) as discussed earlier. The JSDT provides a high level interface for use by developers of collaborative applications (among others) which is independent of the underlying communication protocol. That is, calls to the JSDT API do not specify or presume a particular protocol[3] and therefore do not need to be changed if one protocol is replaced by another. As a result, one can use the JSDT with any protocol that allows data to be exchanged among a number of collaborating entities. In its current distribution (version 2.0), the JSDT supports a TCP or UDP sockets protocol which in theory could be used for both LAN, WAN and Internet traffic. However, in practice, many firewalls will block attempts to make a direct TCP connection between two machines and so the JSDT also supports an HTTP protocol which can "tunnel" through such firewalls provided the server side is properly configured. This is why PeerView allows the user to specify either protocol. The user must also specify the port to use as target when packets are routed through the network, and she must make sure that the firewall on both the server and client side allow traffic to pass through this port.

Sockets based traffic is a simple form of data exchange and for collaborative applications, IP multicast is a potentially better solution. Whereas unicasting data as done in TCP communication requires the sender to send separate packages to each receiver, a multicast protocol allows the sender to send just one package and then have it replicated by multicast routers on the network and forwarded to the intended recipients, possibly through several more replications. PeerView does not support multicasting in its present form because it is primarily intended for work environments where the data density, i.e. the amount of traffic generated, is moderate and not sufficient to "swamp" the network. However, an application that must scale to more demanding environments will probably need to use multicasting protocols as data traffic increases to achieve acceptable performance on all but dedicated networks.

As mentioned, JSDT can be made to work with any suitable protocol implementation, according to its documentation, but for multicast there is fortunately a way to minimize time consuming experimentation. It is, once again according to JSDT documentation, possible to couple the JSDT with the LRMP (Light weight Reliable Multicast Protocol) package from Inria [39] with no great difficulty. The LRMP implements a true push service meaning that clients need not be aware of the server's name nor have a port number to connect to. Instead, they subscribe to an information channel and receive data when it is broadcast, so the distribution scheme resembles a radio channel rather than an telephone line as is the case with the sockets channels. In addition, the LRMP architecture is based on IP multicast with its associated advantages of supporting both one-to-one and one-to-many communication as well as making effective use of available bandwidth.

The primary reason PeerView was not implemented using the LRMP protocol was my reticence towards using a technology that I had no implementation ("hands-on") experience with. Sockets based communication is the default protocol in JSDT and consequently the best documented. Also, I was familiar with the basics of sockets programming and so felt more comfortably using that technology, knowing there were deadlines to be met for the PeerView project. In retrospect, I am not sure this was the right decision because the advantages of the

[3]The protocol may be configured via the JSDT API, but that does not detract from the orthogonality of the API as a whole.

LRMP (greater scalability, simpler user interface, more functionality) may have outweighed the cost in increased development time. It is, of course, difficult to assess since I did not experiment with the LRMP protocol and therefore have no experience to base an evaluation on. However, for any future project, InSiter in particular, a multicast implementation would be preferable to a sockets based implementation because the advantages of the former become so compelling as the scope of the application increases that it would be indefensible to use a simple unicast solution.

PeerView uses only a subset of the functionality in the JSDT library, namely the data distribution mechanism. The library also has support for data sharing which allows a group of clients to share a data reservoir (implemented as a byte array) and for implementing a security layer which enables controlled access to shared data and data distribution. This limited use of available functionality is not merely due to the limited scope of the PeerView application. I suspect that most collaborative applications, InSiter included, will have a greater need for a scalable, efficient, reliable data distribution layer than for auxiliary features such as data sharing, security and the like. The reason is that with a sound data distribution layer such as LRMP multicasting in place it is relatively straightforward to implement features such as security (for example by maintaining a registry of clients and their access privileges) and data sharing (by using a byte array and a simple token based access scheme) on top of that layer. If this is the case, then much of the JSDT functionality is unnecessary for the implementation of InSiter, although the JSDT is far simpler than the alternatives surveyed and mentioned in Section 3.1. Instead, the best solution may be a library aimed purely at data distribution and therefore optimized for that purpose. The Web Canal [39] tool from the French research institute Inria seems a good choice because it uses the LRMP protocol also from Inria and therefore has the desirable properties of that protocol as discussed earlier. Furthermore, Web Canal is freely available and pure Java so neither licensing nor compatibility concerns need hinder or limit integration into one's own applications.

The experience gained from implementing the PeerView distribution layer is perhaps the most valuable implementation lesson to be derived from the entire project because the choice of data communication library has such a significant impact on both the internals of the implementation and the end user experience (by influencing network load, responsiveness and ease-of-use).

Chapter 6

InSiter

6.1 Introduction

In this Chapter, I present a tentative design of a system for rendering and review that builds on the experience garnered from the development and evaluation of PeerView. This system is considerably more ambitious than PeerView and is intended for practical, day-to-day use in a realistic setting, be it in academia or industry. The working title of this system is *InSiter*, a name chosen to signal the organization-wide scope of the rendering it presents to its users. In the following Sections, I describe the intentions and design philosophy of InSiter and present the interface design of its main components, and discuss in general terms how this could be implemented.

6.2 User audience

The intended users of InSiter are the same as for PeerView, namely first and foremost software developers, but as with PeerView, InSiter should be designed and implemented so that is accessible to most information workers (from office clerks to civil engineers) without extensive training. I believe this is not only feasible, but also desirable. Not just because it widens the potential user audience, but also because it forces design work to focus on the essentials of the product and how they are best communicated to all users. In Section 3.2.1, I described the principles of simplicity in connection with interface design and how they are propounded by industry and academia pundits. This credo of the simple and rejection of the overly elaborate[1] recognizes the inherent limitations of the human cognitive system and, arguably, has some intrinsic aesthetic appeal (see also [55]). The InSiter design should therefore cater to its intended audience, i.e. software developers, but at the same time be as inclusive as possible.

6.3 Design objectives

PeerView was intended as a prototype to test the viability of the ideas of rendering and review, i.e. both the usefulness of a design philosophy based on those concepts and the feasibility of

[1] Facetiously but incisively referred to as the "keep it simple, stupid" or KISS principle by some.

translating them into reality. It did yield useful information on the latter aspect, but little on the first since it was, by necessity, limited in scope and therefore did not lend itself to the type of prolonged experimentation in realistic settings needed to make conclusive statements on usefulness.

InSiter should not be an experimental system — it is intended for "production use", but it might still produce valuable experimental results by allowing the actual usefulness of rendering and review to come before a day, so to speak, by being used by real groups working on real projects in real organizations. The purpose is therefore to produce a workable (in the sense of being accessible and useful) and working (in the sense of being well designed and implemented) rendering and review system that builds on the experience from PeerView to provide users with a common information space and facilities for reviewing and discussing its contents.

The InSiter interface will be a ZUI (zoomable user interface) much in the same way as the PeerView interface, but it will employ a wider range of ZUI techniques because it must allow users to insert more types of artifact into its interface and allow them to apply a greater and more diverse set of operations to those artifacts than PeerView did. This is necessary to give users the freedom to customize the information space to suit their needs.

The PeerView interface was *document-centric* in that the only type of artifact that could be placed on its scalable desktop was documents, albeit their contents ranged from bitmaps to HTML code. The InSiter interface retains this emphasis on documents since such finite strings of symbols are after all the only artifact that information workers can produce, but it widens the scope to include the relationships between those artifacts, the people who create them and the organizational relationships between those people. By doing so, InSiter aims to render not only the outcome of a project but also the context it which it is brought about.

InSiter will allow users to form collaborative units (which will be referred to as groups for the time being) and will give them more extensive communication and review facilities than PeerView did. It will support a live video feed of each group member to be placed in the panorama so as to provide a video mosaic similar to that discussed in [17]. Further, InSiter will allow members to communicate directly on a member-to-member (unicasting) or member-to-group (broadcasting) basis by exchanging text messages which can contain hyperlinks to artifacts in the common information space and thus used as a simple means of initiating collaboration on a specific artifact. InSiter will give users the opportunity to discuss individual document artifacts by providing review facilities similar to what is found in PeerView. There seems to be little need to introduce new review facilities since the discussion forum concept in PeerView captures most of what transspires in a simple code review session which is what inspired it. Usability testings of early version of InSiter can probably reveal whether more sophisticated review facilities are desirable and so I leave those considerations to a later, more advanced stage of design.

InSiter will generate and store statistical information on document artifacts to facilitate analysis of them.

6.4 Interface design

6.4.1 Introduction

One of the most difficult design decisions is where to begin. After all, there is nothing to hinder design from beginning with how to ship the product to end users before describing how architecture and interface should be designed. However, the interface is a good place to start because it constitutes a concrete, visible commitment that can be understood by both end users and developers, and therefore sets a lower threshold for what subsequent layers of functionality and architecture must support.

Figures 6.1 to 6.6 are mock-up screen designs showing how the InSiter panorama will appear to the user. This is a simplified representation that captures the overall intent, meaning that details may differ in the final design. Based on the experience from PeerView and the experimental evaluation described in Chapter 4, I believe the exact nature of those details could favourably be decided on through repeated, i.e. iterative, evaluation of progressively refined design proposals. That way, the final design would likely be an accurate reflection of user needs, although the rules of thumb applied in the design of the PeerView interface (see section 3.2.1) can also help guide interface design.

However, for any evaluation to take place, there has to be some interface in place to begin with, and the below screen designs serve that purpose. Section 6.4.3 then describes the rationale behind these designs.

6.4.2 Screen designs

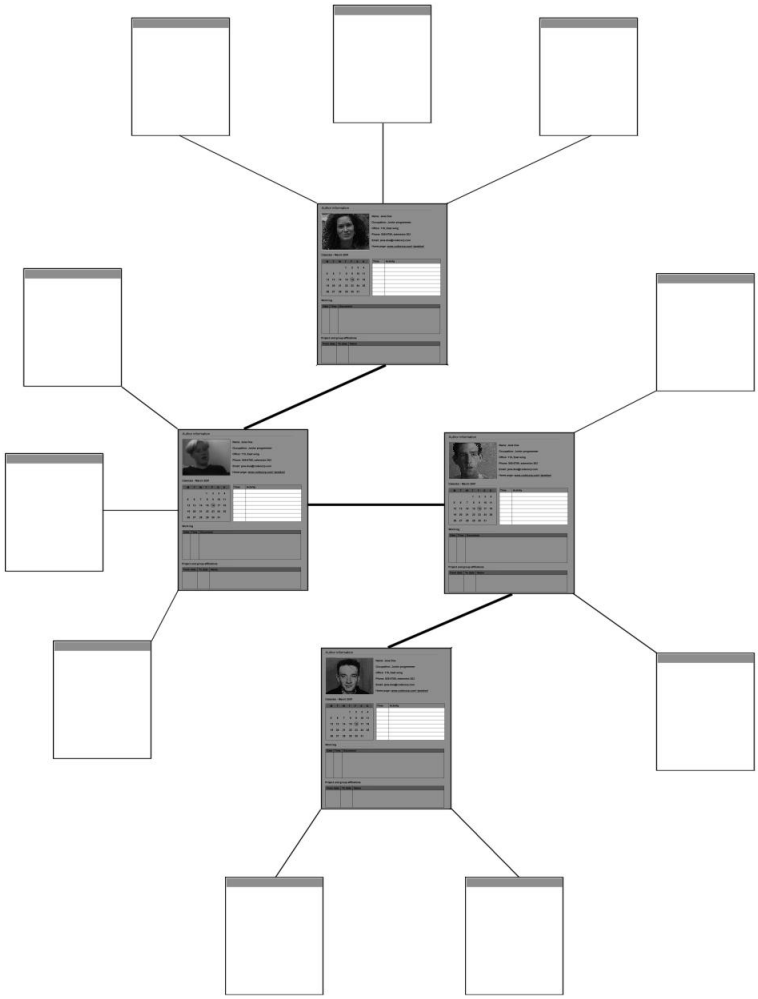


Figure 6.1: A mock-up of an InSiter panorama. Four author nodes are connected to a number of document nodes, all of which can be translated and scaled individually by the user. A variety of layout managers can be used to automatically lay out nodes so that they are arranged by author, by modification, by some geometric arrangement (as with the grid layout manager in PeerView), or by some other criterion entirely. The user can zoom manually or by double-clicking any node or edge.

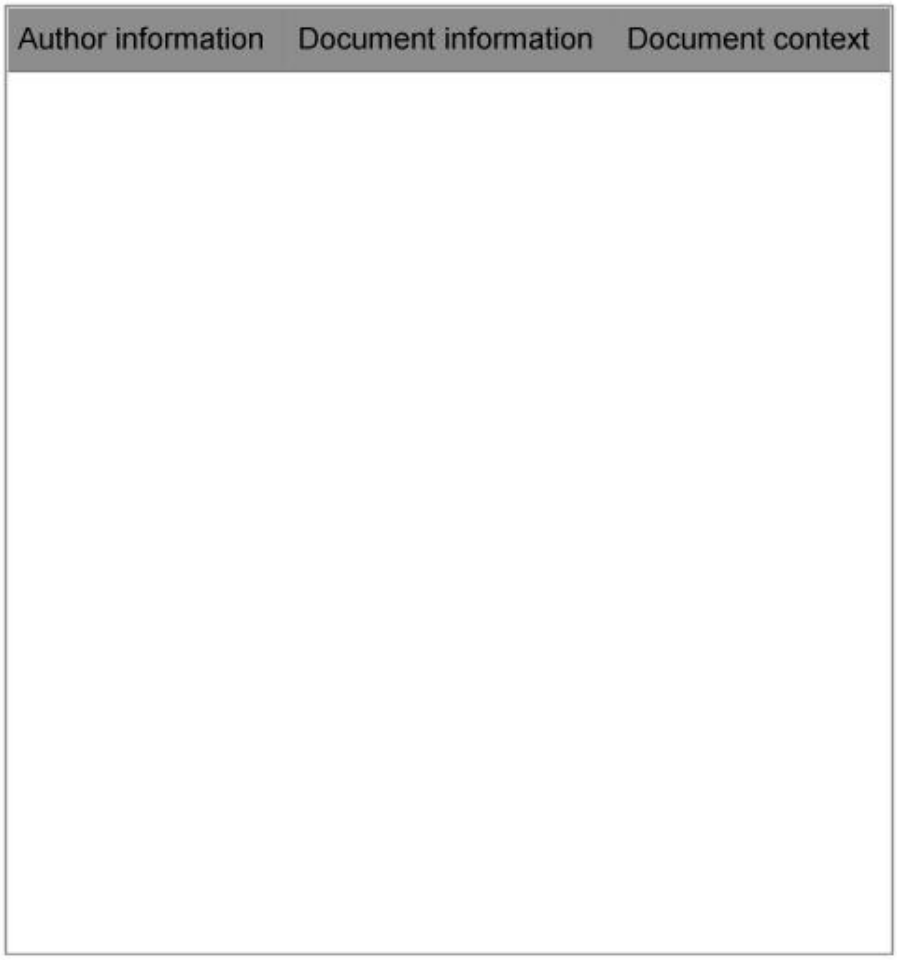


Figure 6.2: Centered view of InSiter document. The user can double-click each of the three text items in the header bar to zoom quickly to a centered view of either one.

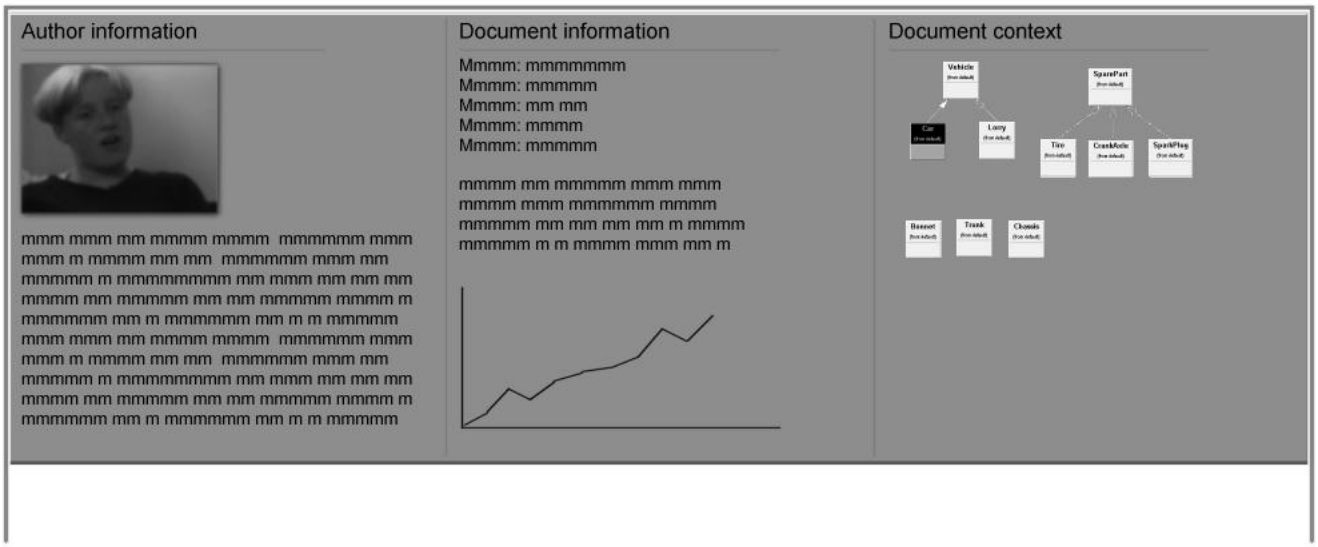


Figure 6.3: Close-up view of InSiter document. The user has manually zoomed to a magnification that allows her to see all three portions of the document header bar. From here, she can double-click any of them to have it centered automatically or continue the manual navigation.

Author information

Name: Jane Doe

Occupation: Junior programmer

Office: 115, East wing

Phone: 555-6789, extension 303

Email: jane.doe@codecorp.com

Home page: www.codecorp.com/~janedoe/

Calendar - March 2001

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Time	Activity

Work log

Date	Time	Document

Project and group affiliations

From date	To date	Name

Figure 6.4: Author information page. This portion of the InSiter document header bar contains information on the author of the document and a live video feed of her (if available).

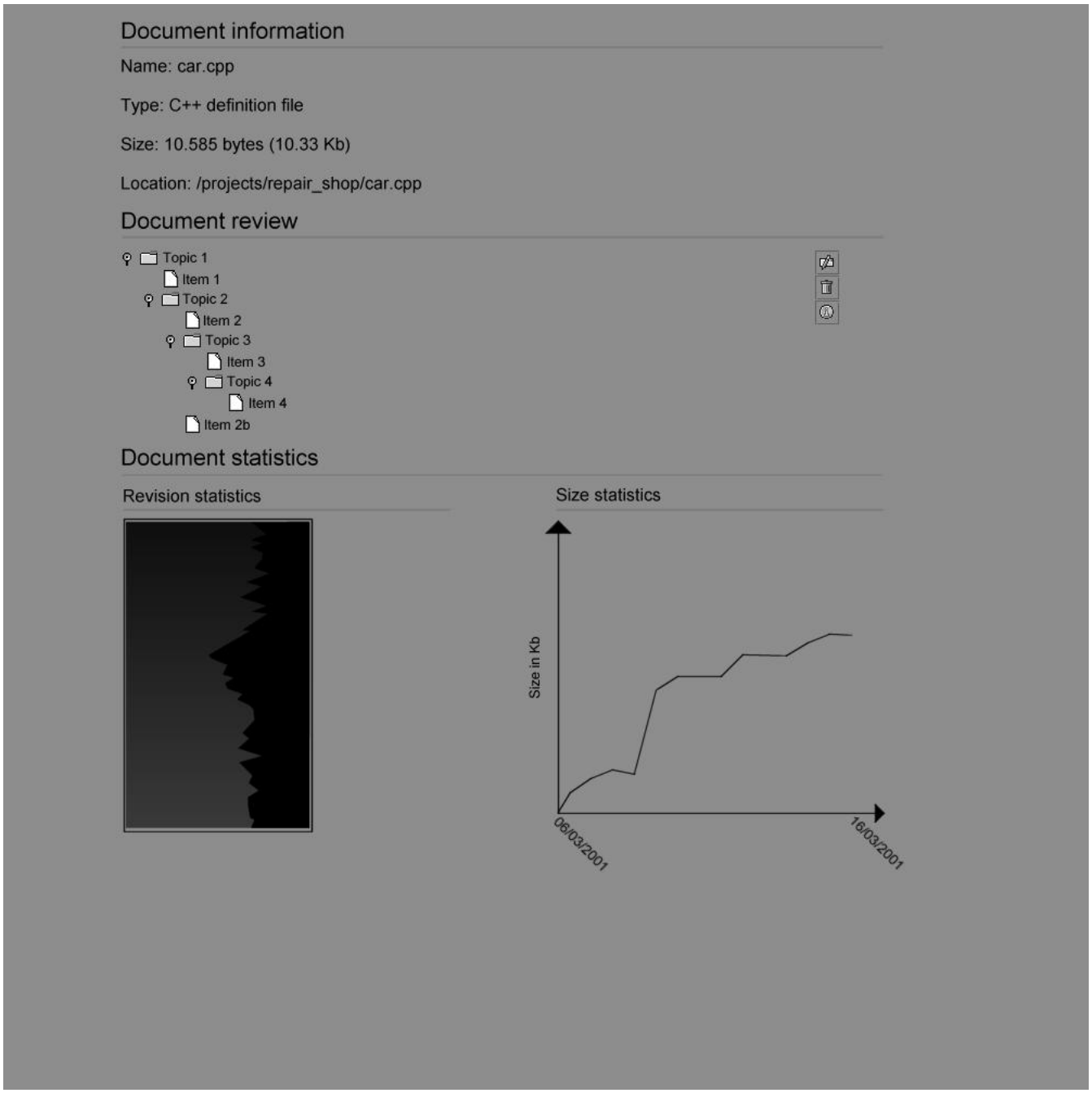


Figure 6.5: Document information page. This portion of the InSiter document header bar contains summary information on the document and a revision diagram (lower left) that shows an outline of the document along with an indication of when each line was last modified (colour coded with red indicating recently modified and blue the opposite).

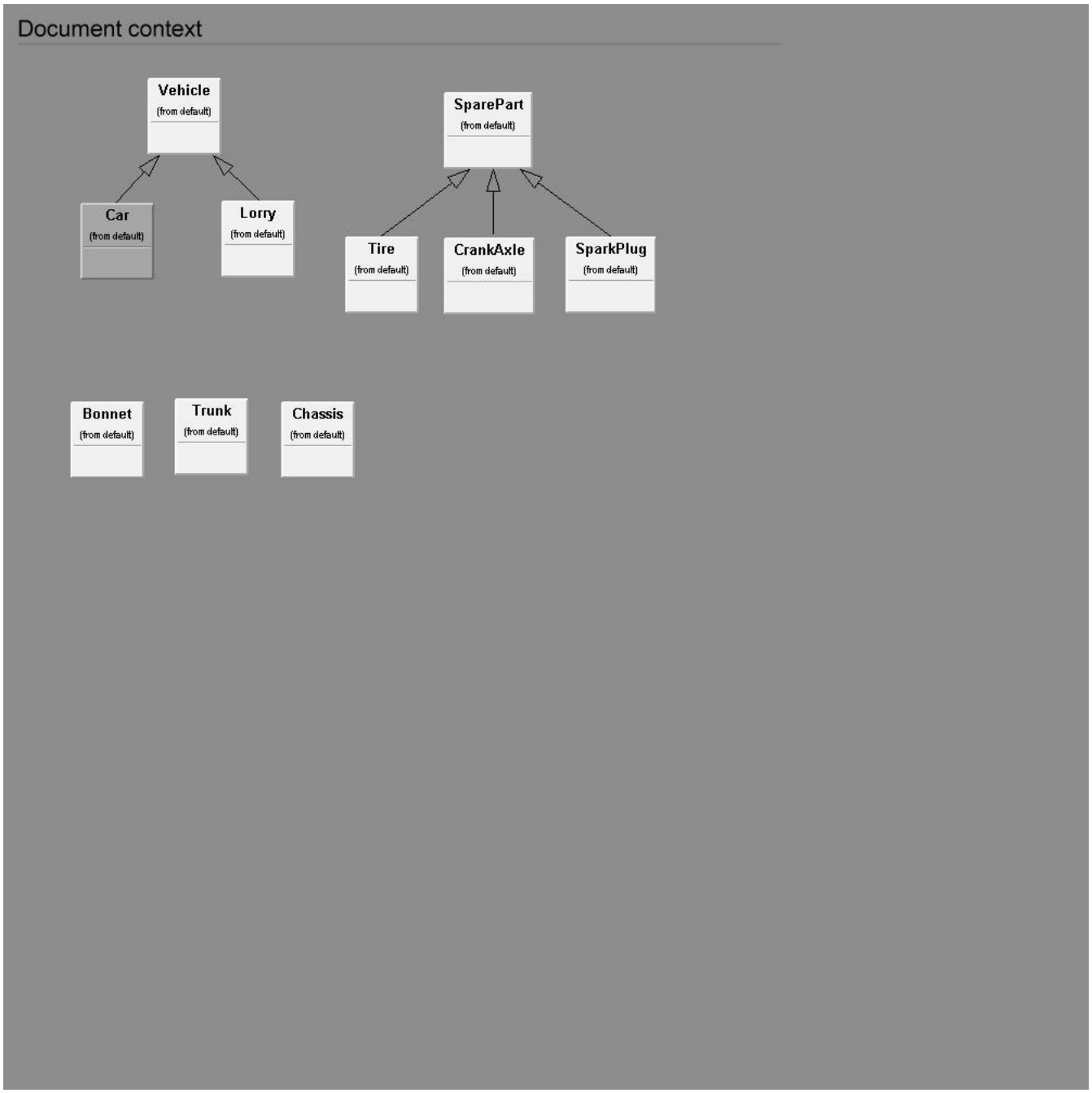


Figure 6.6: Document context page. This portion of the InSiter document header bar contains a user customizable representation of how the document relates to other artifacts. In this case, the document in question is a code file and its context is therefore appropriately represented by a UML class diagram.

6.4.3 Design rationale

InSiter aims to provide users with a comprehensive common information space as is reflected by the above screen designs. Section 6.5 describes the overall architecture that must support

the proposed designs, but the rationale behind the inclusion of such features as video feeds and code statistics merits separate attention.

Video broadcast

In [17], Lee et al. describe their experience with a system called Portholes "that allows distributed work groups to access information related to general and peripheral awareness" [17, p. 385] and report a mixed user reaction. However, they also find that user reticence was attributable to such factors as camera shyness, threat of surveillance, loss of privacy and lack of control, all of which they address with varying degrees of success by allowing users to edit, blur and remove frames from the video feed streaming from their camera. They also found that users request a "lookback" facility for determining who is accessing their video feed or has done so recently. In other words, users want to be aware of who is looking at them so that they do not feel passively monitored, but rather on par with whomever is observing them. User control and support for customization is therefore vital if the InSiter video broadcast system is to be successful. Portholes provides this through a preferences section for each user where he or she can regulate who has access to which images and how those images appear. I plan to incorporate a similar facility in InSiter, but the specifics of the design will depend on the type and quality of video used. However, users should in any case be able to edit out specific frames, i.e. block the image stream at will and at any point, and should be able to see at a glance who is currently looking at them (this reciprocal relationship between observer and the observed is sometimes referred to as "reciprocity").

As for the technical feasibility of implementing a video broadcast system, consider the following back-of-the-envelope calculations:

Notation:

f the size in bytes of each broadcast frame.

r the frame rate measured in number of frames broadcast per second.

c $\frac{1}{compressionfactor}$.

b the bandwidth in bytes pr. second necessary to broadcast $f * r * c$ bytes.

Assume:

- A resolution of 320 by 200 pixels and a pixel depth of 8 bits (sufficient for 256 colours or grey scale). Then $f = 320 * 200 * 1 = 64.000$.
- A reasonable frame rate of $\frac{1}{3}$ the real-time frame rate. Then $r = 8$.
- A 1:10 compression ratio which is a conservative estimate, judging from [29]. Then $c = \frac{1}{10}$.

Then:
$b = 64.000 * 8 * \frac{1}{10} = 51.200$ bytes = 50KB

Notice that this estimate is independent of the number of users in a group if IP multicast technology is used, as discussed in Section 6.5.2, since multicast allows the sender to broadcast to multiple recipients without multiple transmissions by instead making a single transmission to a group address which is then automically forwarded to the members of the group in question. 50KB is a fraction of the bandwidth available in standard local area networks and within the range of many private subscribers with the increasing availability of ISDN and ADSL services. InSiter will need to distribute other forms of data, but video will probably account for a significant if not dominant portion of the total bandwidth consumption and so the above is also an estimate of the magnitude of the total InSiter bandwidth consumption.

Code statistics

In [52, p. 315 ff.], Eick describes the *SeeSoft* system which translates large amounts of code into a visual representation consisting of long vertical columns corresponding to the files containing the code in question. Each column can show the contours of a portion of code, i.e. how it would appear if viewed at a distance or, equivalently and more pertinent to InSiter, from a high elevation (that is, zoomed out so there is significant difference between the z coordinate of the viewport and that of the code). Each pixel line in a column then corresponded to a line of code and was colour coded to indicate its most recent date of revision. A similar arrangement is used in Figure 6.5 where an outline of the entire document is shown in the lower left corner. Each line is coloured so as to indicate when it was last changed and the user can double-click any portion of the outline which will result in InSiter zooming out to an appropriate viewing distance and scrolling to the portion of text corresponding to the clicked area. This feature will most likely only be enabled for documents containing computer source code since its usefulness lies in the fact that formatted source code has a distinctive signature outline that enables experienced programmers to quickly identify relevant segments of code by inspecting that outline [52, p. 318]. Also, to gather the necessary revision statistics, InSiter must either interface with the development tools used to create the documents in question, or make a line-by-line comparison of successive file versions each time a document is updated. The latter is a better solution since it does not depend on any interfacing with the constantly changing and largely proprietary development tools in use today. It does, however, carry with it a processing cost and should therefore only be done where it yields a useful outcome. The information thus gathered can also be used to implement a special inspection mode that temporarily modifies the apperance of document artifacts to give a SeeSoft-like overview representation. This inspection would expand each document so that all of its contents was visible without scrolling and then arrange the documents beside each other so that they made up a linear range of columns. By scaling to an appropriate level and perhaps activating a optional colour coding to indicate revision history, the user could then use the resulting arrangement of artifacts in the same manner as SeeSoft.

Again, computer source code files are the type of document for which such an overview representation would make most sense because of the relationship between code formatting and semantics (if-then statements and switch-case clauses in the *C* language are usually indented

in an easily identifiable fashion, for example). However, if the line-by-line comparison scheme suggested above was used, it would be possible to maintain revision statistics for other text document types as well. Future design refinements will have to address this question in detail.

The other type of statistical data envisaged in this design is the size evolution graph in the lower right portion of Figure 6.5. This is included because it gives a useful, if incomplete sense of how the document has evolved over time while at the same time indicating when it was created and what its current size without taking up significantly more space than those vital data alone would have taken. Also, it is trivial to implement since the size of each document will be monitored anyway in the process of maintaining updated copies of the artifacts placed in the InSiter panorama.

Document context information

In Figure 6.6, the example document used is shown in context using a UML diagram. This is an illustration of the general principle that when users add documents to the InSiter panorama, they can specify the relationships of those documents to other documents. They should be able to do this either by attaching a separate document containg the necessary information (such as a UML diagram in some portable format) or by entering the relationship textually or using a mouse. InSiter can then use this specification to render a document context pane as show in figure 6.6. This need not be a UML diagram, but can be any representation that can be rendered in the InSiter panorama such as a dependency graph connected to a document showing which other documents must be updated when that document is. InSiter could then be configured to automatically notify affected parties when a document is changed that other documents needs to be maintained in response to the change. All of the above seem to pose no insurmountable technical problems: the specification of relationships can be entered using standard interface widgets, diagrams can be constructed using simple graphical primitives and responding to changes in documents is can be reduced to a question of traversing a dependency graph, if one exists and submitting messages to any users affected. I therefore leave the details of this design to a later stage.

6.5 Architecture

In [10], Booch, Rumbaugh and Jacobson (the so-called "three amigos") define software architecture so:

> "An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition."

This abstract formulation captures what is probably a common intuition about software architecture, namely that it has to do with the relationships between the components that make

up a software system, and it does so in a way that stresses the engineering aspect of software architecture. But architecture can be interpreted differently, namely as the "style and method of design and construction [of a complex system]" [5], which is a definition closer to how conventional architecture can be perceived, namely as the consistent and reasoned application of design principles to achieve the most desirable balance between what is aesthetically pleasing, functionally required and economically feasible. I see the job of the software architect as much the same as that of the traditional architect, i.e. a person who design structures to be implemented by engineers and used by whomever contracted them or otherwise has use for them. The InSiter architecture is therefore designed to accomodate the needs of both parties by being:

- As simple as possible, but no simpler.
- Open and extensible.
- Flexible to allow for user customization and easy redesign at later stages.
- Robust, reliable, efficient and secure, in that order of significance.

6.5.1 Information architecture

I use the term "information architecture" to signify the choice and design of data structures to represent the information and interface components that the user can access through the InSiter interface. It is assumed that all data structures in the finished program correspond to interface components, although some may do so indirectly, perhaps by serving some auxiliary purpose in relation to another structure. The below only addresses the information architecture issues in InSiter that are either new, i.e. not addressed by PeerView, or of particular interest because they affect the entire design or pose some form of difficulty.

The information space multigraph

The most complex portion of the InSiter interface is the panorama where artifacts are placed and manipulated. The collection of artifacts visible in the panorama constitute a *multigraph*, i.e. a graph structure where pairs of vertices may be connected by multiple edges rather than a unique edge [2]. The vertices correspond to either *author nodes* or *document nodes* in the InSiter panorama and the edges correspond to relationships between authors, between documents or between author and document(s) [3]. Relationships may be either unilateral or bilateral and the direction of an edge between a pair of vertices indicates which apply in a given case. An example is given in Figure 6.7 where a set of author nodes are indicated by vertices marked An and a set of document nodes are indicated by vertices marked dn (with n being an integer).

[2] A multigraph can be defined as a pair $MG = (V, E)$ where $V \neq \emptyset$ is a finite set of vertices and E is a set of two-element subsets of V, where any $v \in V$ may be in more than one $e \in E$

[3] The terms vertex and node will be used interchangeable throughout this Chapter.

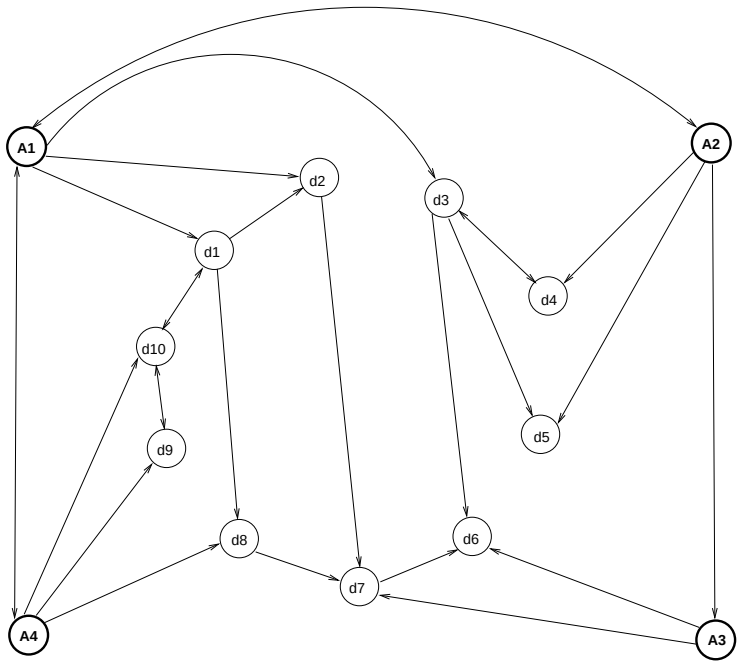


Figure 6.7: An example of an information space multigraph

One interpretation of the graph in Figure 6.7 could be that author $A1$[4] is the current maintainer of documents $\{d1, d2, d3\}$ and that authors $\{A2, A4\}$ are her co-workers. Similarly, author $A3$ is responsible for documents $\{d6, d7\}$, but she does not have a bilateral relationship to any of the other authors, and the unilateral relationship to author $A2$ is most likely an indication that $A2$ is her superior in the organization to which they all belong. The relationship between author and document is always unilateral and therefore represented by an edge directed from the author node to the document node. Documents, on the other hand, may be bilaterally or unilaterally related, as appears from Figure 6.7. A unilateral relationship may indicate a one-way dependency such as that between a `C++` code file and a standard library, or it may indicate some other, user-specified relationship such as that between a legal text and its annotations or a template and the documents based on it. A bilateral relationship may indicate a two-way dependency such as what exists between a `C++` definition file (typically suffixed by `.cpp`) and the corresponding declaration file (suffixed by `.h`), or it may indicate an entirely different bilateral relation between documents.

The multigraph structure described above is an abstract representation of what can be found in the InSiter panorama. The user will not need to be aware of this structure when placing artifacts in the panorama, although she might be, but it will be the underlying structure behind the artifacts visible in the InSiter interface. The concrete meaning of the multigraph will depend on how it is *annotated*, i.e. what properties the individual vertices and edges are "decorated" with as users collaborate and add artifacts to the panorama. For instance, the graph in Figure 6.7 is an abstract representation of the panorama that results when 4 authors collaborate in a group on a collection of 10 documents in all, but it says nothing about either the rendering of nodes or the rendering of relations between them. Let us assume that author $A1$ has added one text document containing HTML code and two bitmap graphics documents to the panorama, represented by nodes $d1$ and $d2, d3$, respectively. Let us further assume that

[4]Rather, the author designated by the node labelled $A1$, but I will dispense with such verbiage and simply refer to nodes as authors and edges as relationships where appropriate.

she has indicated that node $d1$ is related to nodes $d2, d3$ by a dependency relation since the latter two are bitmaps that are referenced by the HTML code in the former. Let us also assume that when $A1$ joined the group to which she currently belongs, she not only submitted her own profile to the group, but also stated her relationship with the other members, i.e. authors $A2, A3, A4$ in a similar manner to how she indicated the relationship between documents. We then have two sets of information that can be used to determine the rendering of both documents and author nodes as well the relationships between them. This gives rise to an *annotated multigraph* which can be defined as a set of five sets $AMG = \{V, E, A, Av, Ae\}$, where $V \neq \emptyset$ is a set of vertices, E a set of two-element subsets of V, A is a set of annotations, $Av = \{(v, a) \in Ae : v \in V \wedge a \in A\}$ and $Ae = \{(e, a) : e \in E \wedge a \in A\}$. An annotated multigraph is a both structural and databased representation of the information space that an InSiter panorama constitutes and therefore seems an appropriate data structure on which to base its design. This argument is made stronger by the added benefits it brings with it, namely a large body of research on graph related issues and algorithms and much existing code for implementing such structures.

Of course, as with any graph structure, the abstract, set theoretic definition above says nothing about how it should be embedded in the plane, or, in the case of a zoomable display, in three-dimensional space. This means that there is no inherent correlation between the abstract multigraph representation and the placement of the corresponding artifacts in the panorama. An illustration of this is in the relationship between Figures 6.1, 6.8 and 6.7. Figure 6.1 shows how the InSiter panorama might appear (these are tentative design plans, so final versions may differ) after four users have joined a group and added artifacts to the panorama. The number of nodes and relationships between them (indicated by lines) correspond to the number of vertices and edges in Figure 6.7, but there is no deliberate correlation between the (arbitrary) layout in Figure 6.7 and the deliberate layout in Figure 6.1. Figure 6.8 shows the relation between the two other Figures more clearly. It is a diagrammatic representation of the panorama where attributes have been removed to indicate only the graph structure underlying the panorama contents. Neither Figure 6.1 nor Figure 6.8 contain edges between documents as this would obscure these particular illustrations, but the intention is that users should be able to regulate freely which types of nodes and relationships should be rendered in the panorama. The absence of edges indicating relationships between documents could therefore be interpreted as the user having requested that such edges not be drawn. Section 6.4 has more on this topic and also shows how document relationships are shown elsewhere in the panorama, namely in the "Document context" area of individual document nodes.

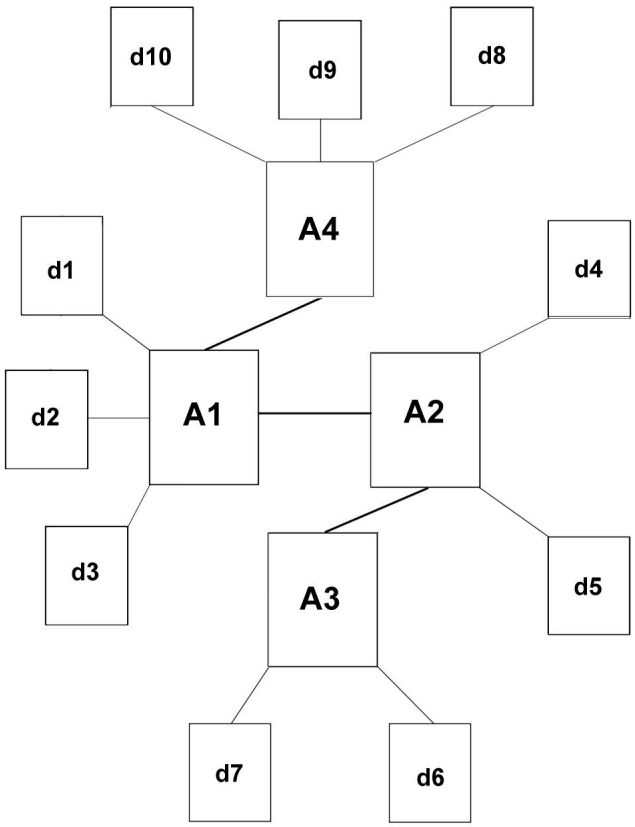


Figure 6.8: A diagrammatic representation of the panorama in Figure 6.1 to show its relationship to the multigraph in Figure 6.7. Note that the physical layout, which may have been applied manually by the user or automatically by a layout manager, has no deliberate relation to the layout of the graph in Figure 6.7 since a graph, by definition, has no inherent geometric embedding.

The multigraph structure will be represented by a modified version of the scenegraph structure that is used in the Jazz library as discussed in Section 3.3. Nodes in a Jazz scenegraph can be either leaves or inner nodes, both of which can be decorated with arbitrarily complex visual components which is the mechanism used for storing documents in PeerView. This corresponds to the annotations of the multigraph discussed above, and the relationships between nodes can be implemented by wrapping leaf nodes in a simple structure containing a collection of pointers to other leaves (or inner nodes) and tailoring algorithms that are either built from the ground up or taken from a third party library to work with this modified structure. Since the Jazz scenegraph supports many different types of operation, from translation, scaling and rotation to invisibility, hyperlinking and fisheye deformation, such a modified structure will also be endowed with a host of sophisticated graphics facilities at little or no extra cost in development time. The details of this design are not in place at the time of writing, but as the above suggests, it seems a both feasible and promising implementation model.

Data storage

InSiter will generate and need to store persistently (on disk) or transiently (in memory) the following types of data:

- Review contributions — these are plain text contributions by individual authors as well as the meta-data needed to identify and classify them (author name, date of creation and so forth).
- Review discussion structure — these are structural descriptions of the relationships between review contributions. They can converted into a textual representation since contributions are identified by their meta-data.
- Panorama structure — the multigraph defining the contents of the InSiter panorama can be converted to a pure text form ("serialized", in Java parlance) and hence easily stored in a text file.
- Document artifact statistics — as appears from Figure 6.5, InSiter will maintain statistical information on each document and since such statistics span the lifetime of the document, it will have to be committed to persistent storage between invocations of PeerView.
- Author information — this does not include the live video feeds of authors, but only encompasses their personal and other information as shown in Figure 6.4 which can all be converted to a pure text representation and thus easily saved to disk.
- Group data — the collaborative units that authors can form (referred to as groups at this early stage of design) are represented by textual data specifying name, description, participants and so forth, all of which can be stored as pure text.

As is suggested by the above description, I plan to store as much data as possible in a pure text form by using the Java serialization mechanism which allows data structures that implement a specific interface (the `Serializable` interface) to be converted to a textual representation and to subsequently be restored from one. By doing so, InSiter can commit all of the above data to disk using the standard stream facilities found in the JDK [50] when the application is terminated, and restore them from disk when it is subsequently restarted. This approach is used in PeerView which saves discussion fora, discussion contributions, group directories and preferences to disk as serialized representations which is a simple and compact representations that can be manually inspected using standard text processing tools. This might be useful if scouring the materials generated for a specific snippet of information such as a remark in a discussion contribution. InSiter will reap those benefits by using the text representation discussed above, and it can improve on PeerView's storage method by using (optional) compression of files above a certain size which will of course make the text files inaccessible to simple manual inspection, but will minimize the InSiter disk "footprint"[5] if so desired.

Further elaboration of the design might make it necessary to introduce new data structures, but I intend to make them serializable as well so as to not introduce exceptions to the above scheme. I expect little difficulty in doing so since any data structure can be described as a

[5]The amount of storage allocated.

collection of elements and relations between them[6] which ought to be amenable to textual representation.

6.5.2 Distributed architecture

As described in Section 5.3.4, the Light-weight Reliable Multicast Protocol (LRMP) package from the French research institute INRIA seems a promising candidate for implementation of a multicast distribution architecture which in turn seems a good choice of such an architecture for colloborative systems. Indeed, in [37], Liao reports that the LRMP has been used in such applications previously and further describes the design goals of the LRMP library which have been to deliver a reliable protocol for bulk transfer of data, precisely what is needed in InSiter's case. The LRMP is a transport protocol meaning that it implements the transport layer of a network protocol and therefore does not aim to support what is typically left to layers above the transport layer, namely authentication and session management services. This need not be a problem because the LRMP can be used in conjunction with the *WebCanal* [39] package, also from INRIA, which provides high-level primitives for creating and maintaining channels that can be used as communication conduits by the colloborative units. An alternative solution to using the LRMP protocol could be to build on top of the `MulticastSocket` class that is part of the standard JDK [50]. This class offers the bare minimum of services needed to send and receive multicast data, but can be extended using the standard Java extension mechanisms (composition and inheritance) so one could conceivably extend this class to support both managed groups and multiple channels in a manner similar to the JSDT. The advantage of that approach would be that one could customize freely the multicast protocol to suit the particular needs of InSiter which would most likely yield better performance than using a third-party solution aimed at generalized support for applications relying on multicast. Also, it would reduce the potential for complication incurred by interfacing technologies to an absolute minimum since extending a JDK class to fit into the InSiter framework would not constitute a technology interface, but simply a program component like any other.

It therefore seems entirely feasible to implement the InSiter architecture based on a multicast protocol as an implementation can be obtained over the Internet or built from the ground up using tested and well-known technology. Issues of security and reliability will have to be dealt with separately as neither the LRMP or the `Multicastsocket` class have built-in functionality to support those aspects. However, the Java JDK contains several classes for implementing security layers and encryption technology for Java is freely available over the Internet (see for example [13]), so fitting a distributed architecture with customized security and encryption layers seem no more complicated than implementing the multicast transport layer.

6.6 System requirements

InSiter is to be implemented in Java. This is a given if the Jazz library is to be used since interfacing with a different language brings with it its own set of difficulties. I do not expect performance difficulties to hamper usability or development since PeerView has demonstrated

[6]This may sound like a gross generalization, but consider for a moment that a "data structure" by definition is a collection of discrete symbols sets and a collection of relations (which can be defined as pairs of elements) between them, and the statement may seem more plausible.

that both the implementation of a zoomable interface and the distribution of data, even using a non-optimal delta algorithm (see Section 3.3.6), do not offer unsurmountable performance challenges. Nor do I expect the added layer of data incurred by the introduction of video transmission to degrade performance significantly as suggested by the calculation on page 61, since the frame rate can be adjusted according to available bandwidth and compression can be applied to further reduce transmission volumes. The feasibility of the former measure is documented in for example [17], and typical compression ratios of video data can be very good indeed as is demonstrated by the popular video formats AVI and MPEG. I therefore expect the minimum requirements for InSiter to be about the same as those for PeerView, i.e. an average PC system as found in many private homes and in many offices. Each machine should have a network connection in order to participate in groups, and if users wish to use the video facilities, their machines should be equipped with a camera which need not cost more than a few hundred dollars today, depending on resolution and colour range. These modest system requirements, coupled with the choice of Java as implementation language, should make InSiter available to a wide audience and help minimize the deployment difficulties that sometimes arise when new systems are shipped to users.

Chapter 7

Conclusion

7.1 Results and contributions

The work conducted in connection with this thesis has produced the PeerView software which has been the subject of two expositional articles, one quite short which appeared in the February 2001 issue of IEEE DSOnline [61] and another, somewhat longer article coauthored with Jyrki Katajainen which is included in this thesis in Chapter 2.4. The latter article has also appeared as a DIKU technical report [62] and is currently under review by an international journal. PeerView is listed in various directories of applications demonstrating groupware concepts and zoomable interface technology, and the visitor statistics I have gathered for the PeerView home page indicate that it has been frequently accessed by visitors from around the globe, many of whom seem to have downloaded the application.

In the objective terms of public exposure and amounts of code written, PeerView can be said to have fulfilled its purpose of being a prototype vehicle for artifact rendering and group review. Also, the experimental evaluation results and analysis contained in this thesis can likely help successor system(s) be developed more efficiently and to a higher standard than if they had had to be developed without any such empirical base. I believe much of this experience can be of interest also to developers of otherwise unrelated forms of groupware since Chapters 3 and 4 contain information on such matters as distribution technology, design criteria and evaluation techniques that may help developers make appropriate design and implementation choices. An outcome from PeerView that is of potential interest both to groupware developers and to developers building single-user applications is the experience using the Jazz and JSDT component technologies to shorten development time and the deliberations made in choosing them, as discussed in Chapter 3. The PeerView source code is in a sense the definitive reference to anyone who wants to study the details of how these third-party technologies have been used and it has therefore been made public for anyone to peruse.

PeerView has contributed to existing research literature on common information spaces by addressing recurring problems with largely untried techniques, such as using a zoomable display in response to the "detail/overview" problem as discussed in Section 2.2, and a threaded discussion forum for generating and organizing dialogue among users.

7.2 Lessons learned

On a personal level, PeerView has helped me develop my skills as a developer by being a non-trivial development project in territory that was largely untried for me when I started. It has also helped improve my research skills since it spans such a broad range of topics, ranging from theoretical CSCW research on common information spaces [4] to delta algorithms [31] and APIs [50, 34]. The development, research and writing processes have taught me new things about project logistics and confirmed what I already knew or suspected, namely that I tend to be overly optimistic when planning such projects and that as schedules slip, difficult and unpleasant choices have to be made by removing or scaling down planned features and lowering ambitions.

The methodology of building a prototype from an initial idea, experimentally evaluating the end product and performing an analysis of the development efforts to help guide future development proved fruitful in that it did yield useful information and did produce functional software that tested ideas and technologies in practice. Further, the PeerView software provides a "prototyping base" for similar systems since it allows developers to incorporate ideas into an existing framework, i.e. the PeerView source code, and subject them to experimental testing before making end-user versions of them.

This does not mean that the methodology described in Section 2.3 was followed with unswerving loyalty nor that it cannot be improved. For example, as appears from for example the discussion in Section 3.3.6 and 3.3.1, not all decisions were planned for or considered as thoroughly as could be desired. Further, not all phases and steps of the development process took place in the prescribed order. I believe the initial design could have been more stable if I had spent more time studying reports from similar initiatives such as [20] and [7] and drawn the appropriate lessons from the experience reported therein. An example of how this affected the process in practice is reported in Section 3.1 where I only became aware of research on the "detail/overview" display problem after I had carried through my own reasoning on the issue. The experimental evaluation described in Chapter 4 was not planned in detail until after implementation was concluded and the software made public, i.e. quite late in the work process, which resulted in it being limited to the number of participants I could recruit with short notice. In retrospect, I believe that experimental evaluation would ideally be an ongoing series of experiments that punctuated different stages of development and were planned as such, i.e. in detail from the beginning of the project. The first experiment could then be an assessment of the interface design in isolation based on the "mock-up" made during the early stages of design. The second experiment could be conducted when core functionality[1] was implemented and subsequent experiments as additional functionality was added and official releases made. This would of course incur greater costs in terms of time taken to plan experiments, recruit participants and possibly compensating them economically, but the feedback into the development process would make this well worthwhile, I suspect. The techniques used for experimental evaluation worked well in this case and are widely used as documented in [45], but if developers lack the resources to regularly schedule such experiments, they could probably benefit from using a form of heuristic evaluation tailored for groupware [3] for evaluation at the early stages of development.

[1] In PeerView's case, this might adding and removing documents and manipulating the group directory.

7.3 Future work

If possible, I intend to direct future work at completing the design of InSiter which was begun in Chapter 6 through an iterative development process where successive design increments are implemented and the resulting intermediate systems subjected to evaluation which feeds back into the ongoing development process. This would, I believe, be a fruitful methodology for realising a system that could go beyond the PeerView basics to provide users with a useful, scalable and generically designed support system for collaborative work centered around a common information space. However, for this to move beyond wishful thinking there must be sufficient funding in place as InSiter is considerably beyond what can be comfortably be implemented by one or two developers in their spare time. Depending on investor sentiment, InSiter could then become a freely available system for a general user audience or a commercial, licensed system for specific customer groups. Either way, the development process in itself would probably be worth the effort since it can, as demonstrated in this thesis, be a useful asset *per se*.

Appendix A

Design documentation

A.1 Use cases

The below are detailed descriptions of all non-trivial use cases considered as the PeerView interface was being designed. Non-trivial use cases are those that involve multiple branches, iterative user interactions or are central to PeerView by being essential, i.e. indispensable, or likely to be used the most. An example of a non-trivial use case is "Add documents". Trivial use cases are those that involve only simple editing of a set of controls or inspection of a set of values. An example of a trivial use case is "Set preferences". Such use cases are, by and large, entirely predictable narratives and are therefore not included here. Another reason that this distinction is made is to avoid unnecessary elaboration in keeping with the XP (extreme programming) principle of simplicity [6, p. 37]. On a more subjective note, it is common sense to me that work should be done if it is called for by the task at hand, not to enforce ideological adherence to a particular methodology or notational device.

A.1.1 Add documents

Purpose

To add a set of local artifacts to the client application panorama.

Description

1. The client user activates the "Add Documents" dialog box using the relevant toolbar button or menu item.
2. The client application opens the "Add Documents" dialog box.
3. REPEAT until the user has selected all the files he or she wants to add
3.1. The user navigates to a directory using standard controls in the dialog box.
3.2. The user selects one or more files in the active directory.
3.3. The user transfers files to the "Selected documents" box by clicking the appropriate button.
4. The client user clicks the OK button
5. The client application closes the dialog box.

6. FOR EACH selected document
6.1. IF the document is not already in the panorama THEN
6.1.1. The client application adds the document to the panorama
6.2. ELSE
6.2.1. The client application does not add the document to the panorama
7.IF any documents were already in the panorama THEN
7.1. The client application displays an error message.
7.2. The user acknowledges the error message by pressing its OK button.

Variation #1: Removing selected documents

3. REPEAT until the user has removed all the files he or she does not want to add
3.1. The user selects a set of documents from the "Selected documents" box
3.2. The removes the set by clicking the appropriate button.

Comments

The user may close the "Add documents" dialog box at any point by clicking the "Cancel" button.

A.1.2 Remove documents

Purpose

To remove a set of local artifacts from the panorama.

Description

1. The client user activates the "Remove documents" dialog box using the relevant toolbar button or menu item.
2. The client application opens the "Remove documents" dialog box.
3. REPEAT until the user has selected the documents he or she wants to remove
3.1. The user selects a set of documents from the list of documents in the panorama.
3.2. The user transfers the set to the list of documents to be removed..
4. The user clicks the OK button.
5. The client application removes the selected documents from the panorama.

Variation #1: Pruning the list of documents to be removed

3. REPEAT until the user has deleted from the list of documents to be removed those documents that should not be removed after all
3.1. The user selects a set of documents from the list of documents to be removed.
3.2. The user transfers the set to the list of documents in the panorama by clicking the appropriate button.

Comments

The user may close the "Remove documents" dialog box at any point by clicking the "Cancel" button.

Exit application

Set preferences

Customize layout

A.1.3 Open group directory dialog box

Purpose

To open the group directory dialog box

Description

1. The user activates the group directory dialog box by clicking the appropriate toolbar button or selecting the corresponding menu item.
2. The client application opens the group directory dialog box.

A.1.4 Update group directory

Purpose

To update the client side group directory so that its contents are synchronized with the master copy maintained by the server.

Description

1. The client application requests the group directory from the server.
2. The server receives the request and submits a copy of the group directory to the client.
3. The client receives the group directory copy and updates the group directory data structure it maintains.

Variation #1: Request failure

2. The server does not receive the request.
3. The client side connection times out after a fixed period of time.
4. The client application displays an error message in the message area/progress bar.

Variation #2: Submission failure

3. The client does not receive the group directory.
4. The server side connection times out after a fixed period of time.
5. The server application dumps an error message to the console.

Comment

This use case could be refined to handle error states better, but is kept at its current simple form to minimize future work on it. The necessary changes (better error handling) can be introduced at a later stage with little difficulty as they do not affect the structure of the use case.

A.1.5 Submit group directory

Purpose

To update the server side copy of the group directory and have the updated version broadcast to all affected clients.

Description

1. The client application submits a copy of the group directory to the server.
2. The server broadcasts a copy of the new group directory to all clients.
3. The client receives a copy of the new group directory and updates the group directory dialog box.

Variation #1: communication failure

2. The server does not receive a copy of the new group directory.
3. The client times out and notifies the user.

Variation #2: broadcast failure

3. The client does not receive a copy of the new group directory.
4. The server times out.

Comments

The handling of communication failures can be improved, for instance by having the client repeatedly attempting to recommit the group directory copy and have the server do the same for the broadcast of the group directory. However, this would be part of a more elaborate general communication exception handling scheme which seems a questionable investment of resources for a prototype application.

This use case is meant to performed as part of other use cases, but can of course be performed separately.

A.1.6 Create group

Purpose

To create a new group and have it inserted into the group directory.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the "Create group" button.
3. The client application opens the "Create group" box.
4. REPEAT until the group has been given a name and description which do not already appear in combination in the group directory
OR the user presses the "Cancel" button
4.1. The user edits the group name and description fields.
4.2. The user presses the "OK" button.
5. The client application closes the "Create group" dialog box.
6. The client application creates a new group and adds it to the group directory.
7. PERFORM submit group directory

A.1.7 Delete group

Purpose

To delete a group from the group directory listing.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the "Delete group" button.
2.1. IF no group is selected in the group directory listing THEN
2.1.1. The client application displays an error message.
2.1.2. The user dismisses the error message by pressing its "OK" button.
2.1.3. GOTO end of use case
2.2. ELSE
2.2.1. The client application displays a confirmation box requestion acknowledgement of the delete operation.
2.2.2. IF the user acknowledges THEN
2.2.2.1. The client application removes the selected group from the group directory.
2.2.2.2. PERFORM submit group directory
2.2.2. ELSE
2.2.2.1. GOTO end of use case

A.1.8 Edit group

Purpose

To change the metadata of a group, i.e. its name and associated data.

Description

1. IF the group directory dialog box is not open THEN
1.1. PERFORM open group directory dialog box
2. The user clicks the "Edit group" button.
2.1. IF no group is selected in the group directory listing THEN
2.1.1. The client application displays an error message.
2.1.2. The user dismisses the error message by pressing its "OK" button.
2.1.3. GOTO end of use case
2.2. ELSE
2.2.1. The client application displays an edit box with write access to the group's metadata
2.3. REPEAT until the group has been given a name and description which do not already appear in combination in the group directory OR the user presses the "Cancel" button
2.3.1. The user edits the box's contents.
2.3.2. The user clicks the box's OK button.
3. The client application replaces the old group metadata with the new.
4. PERFORM submit group directory.

A.1.9 Join group

Purpose

Description

A.1.10 Select document from visible documents box

Purpose

To select a document from the visible documents box and have it centered in the panorama.

Description

1. IF the visible documents box is not open THEN
1.1. PERFORM open visible documents box
2. The user selects a document name from the listing in the visible documents box
3. The client application makes the name the active selection in the box
4. PERFORM centre document

A.1.11 Zoom to overview

Purpose

To scale the panorama to a magnification where all documents in the panorama are visible and the viewport onto them is at the center of the panorama.

Description

1. The user double-clicks the panorama canvas, i.e. the area outside the documents.
2. The client application computes the magnification M' that allows all artifacts (documents) to be visible in the panorama.
3. REPEAT until the panorama magnification equals M'
3.1. change the panorama magnification by $\frac{M'-M}{n}$ where n is user defined.
3.2. display the panorama at its new magnification

A.1.12 Centre document

Purpose

To bring a document from an initial, uncentred position to a position where it occupies as much of the viewport as possible without distorting its form in any way.

Description

1. The user double-clicks one of the documents in the panorama.
2. The client application computes a position P' and magnification M' that the document must arrive at to be centered.
3. REPEAT until the position of the document equals P' and its magnification equals M'
3.1. change the document magnification by $\frac{M'-M}{n}$ where n is user defined.
3.2. change the document position by $\frac{P'-P}{n}$ where n is user defined.
3.3. display the document at its new magnification and position

A.2 Class diagrams

These are class diagrams of the classes that make up the PeerView 1.0 implementation. They have been divided into three separate diagrams to reflect their division into three Java `packages`. Those packages are: `clientapp` which contains the PeerView client application classes, `serverapp` which contains the server application classes, and `peerviewmisc` which contains classes actually shared by components in the other two as well as classes that currently are not, but are likely to be so at some later stage if development continues. Note that these diagrams reflect the current design, not the initial design which, as discussed in Section 3.2.3 did not contain event handlers and other minor classes that were unlikely to upset the design structure. Such classes were almost exclusively implemented as inner classes, and are not included in the below diagrams because I deemed that they would obfuscate rather than

clarify. Classes containing such inner classes are marked by a small square in their lower right corner and the source code corresponding to such classes can be consulted for further details. A similar argument applies to the choice of connection lines between classes: only inheritance relations (drawn as solid lines) and implements relations (drawn as dashed lines between classes and interfaces) are used.

A.3 Class diagram for package `clientapp`

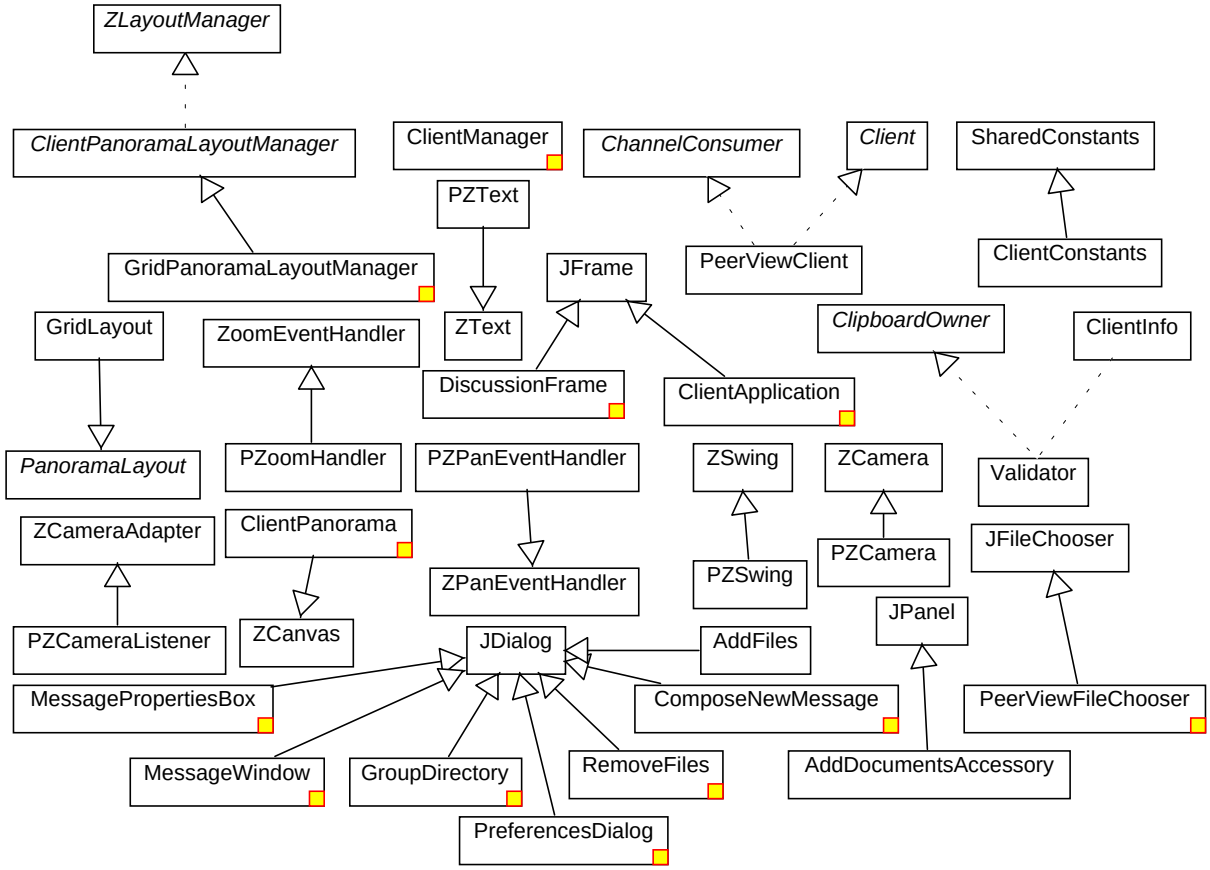


Figure A.1: Class diagram for the PeerView client application classes.

A.4 Class diagram for package `serverapp`

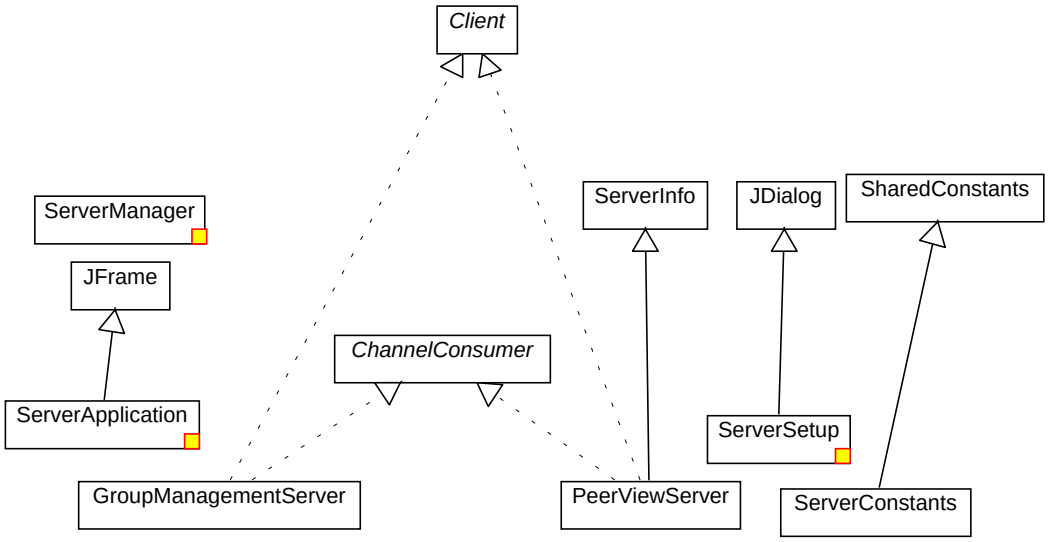


Figure A.2: Class diagram for the PeerView server application classes.

A.5 Class diagram for package `peerviewmisc`

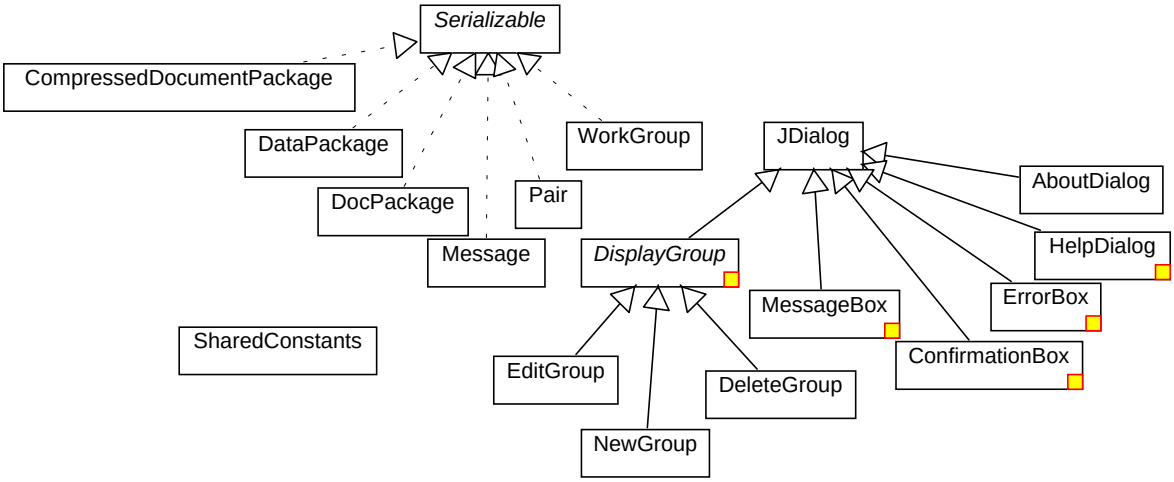


Figure A.3: Class diagram for shared and miscellaneous classes.

Appendix B

Evaluation experiment materials

The below Sections contain the following:

- Reproductions of the task list handed to each participant at the beginning of the experiment.
- A transscript of the audio recording of the evaluation experiment conducted on 14 March 2001.
- Reproductions of the questionnaires filled out afterwards by the participants.

All of the above are provided as copies of the original materials, which were written in Danish, as well as in an English translation for the reader's convenience.

PeerView usability eksperiment – gennemgangsskema 1

PeerView er et prototype system beregnet til understøttelse af samarbejde mellem grupper. Systemet giver mulighed for at placere et antal dokumenter på en delt arbejdsflade og deltage i grupper med andre som har samme mulighed. Derved præsenteres hver deltager for et samlet billede af gruppens arbejdsområde.

Herunder er opstillet en arbejdsgang bestående af en række opgaver udført v.h.a. PeerView hvor meningen er at opgaverne skal gennemføres i den rækkefølge de optræder.
Når du udfører de enkelte opgaver vil jeg bede dig "tænke højt", altså sige (højt og tydeligt) hvis du får et indfald, har en kommentar eller en kritik på et tidspunkt i forløbet. Hvis du f.eks. synes at et vindue er udformet uheldigt eller at systemet reagerer for langsomt, så sig det. Når alle opgaver er fuldført vil jeg udlevere et kort spørgeskema for at få din vurdering af det system du har arbejdet med.

Du kan til enhver tid åbne PeerViews hjælpesystem ved at vælge det fra "Help" menuen. Der kan du få hjælp til de forskellige bokse der vises undervejs i udførslen af opgaverne.

Arbejdsgang (gruppeopretter)

1. Start en netbrowser og hent PeerView client softwaren fra denne web-adresse: http://home.worldonline.dk/~lars_yde/peerview/
2. Installér PeerView softwaren på den maskine du arbejder ved.
3. Find PeerView i systemets start-menu og start programmet.
4. Åbn boksen "Preferences" ved at vælge det dertil hørende menupunkt eller knap.
5. Aktivér fanebladet "Server" og indtast "Ryger 4" i feltet "Server name". I feltet "Server port" skal der stå 4461 og i feltet "Connection type" skal der stå "Socket".
6. Aktivér fanebladet "Personal" og indtast dit navn samt en kort signatur som du ønsker skal tilføjes de indlæg i "debatten" du kommer til at lave senere. Signatur feltet kan lades blankt hvis du ikke har lyst til at indtaste noget.
7. Tryk OK når du sikret dig at fanebladene har ovenstående indhold.
8. Luk PeerView ved at vælge "Exit" fra menuen eller trykke den tilsvarende knap.
9. Start PeerView igen på samme måde som i trin 3.
10. Åbn gruppekataloget ved at vælge "Groups" fra "Setup" menuen.
11. Tilføj en gruppe ved navn "Test gruppe" ved at klikke "Create group" knappen nederst i gruppekatalogvinduet
12. Slut dig til gruppen "Test gruppe" ved at markere den og trykke "Join group" knappen. Luk derefter boksen ved at trykke "Close".
13. Åbn "Add documents" boksen ved at vælge den fra PeerViews menu eller ved at klikke den tilsvarende knap på panelet øverst på skærmen.
14. Tilføj filen "test_gruppe.txt" fra kataloget c:\temp ved flytte den over i boksen "Selected documents". Tryk OK når du har gjort det.
15. Find det menupunkt eller knap der åbner oversigten over grupper og vælg det/den.
16. Åbn boksen "Documents in panorama" øverst i hovedvinduet og vælg dokumentet "test_gruppe.txt". Vent til dokumentet er centreret på skærmen og der er dukket et træ op i nederste del af hovedvinduet.

17. Dette trin er valgfrit – du behøver ikke udføre det. Markér en portion tekst i dokumentet som du kunne tænke dig at give en kommentar til, højre-klik musen og vælg "Copy text" fra den menu der dukker op (pop-up menu).
18. Markér et element i træet i hovedvinduets nederste venstre hjørne. Klik på den knap der lader dig skrive et indlæg ("Compose new message"). Brug hjælpen fra "Help" menuen hvis du har problemer.
19. Skriv en kommentar til dokumentet og indsæt gerne evt. tekst valgt i trin 17 ved at højreklikke i det område hvor du skriver teksten og vælge "Paste text".
20. Afslut boksen ved at vælge "Submit".
21. Dobbeltklik området udenfor dokumenterne i hovedvinduets midte således at der zoomes ud til et oversigtsniveau hvor du kan overskue alle dokumenterne.
22. Højreklik ovenpå et tilfældigt dokument og vælg "Move". Flyt dokumentet til en ny position. Brug hjælpesystemet hvis du har brug for det. Højreklik igen på dokumentet og sæt et kryds i feltet "Lock" hvis det ikke allerede er gjort.
23. Højreklik på et andet dokument og vælg "Resize". Giv dokumentet en ny størrelse.
24. Højreklik på et tilfældigt dokument og vælg "Arrange all".
25. Vælg igen dokumentet "test_gruppe.txt" fra boksen "Documents in panorama".
26. Lav en ændring i filen "test_gruppe.txt" i det notepad vindue der er åbent på din maskine og gem filen. Se om ændringen kommer frem i det dokument der er vist i PeerView.
27. Åbn boksen "Remove documents" og fjern alle dine dokumenter fra systemet. Brug hjælpesystemet hvis du har brug for det.
28. Afslut PeerView ved at vælge "Exit" fra menuen.

Figure B.2: Task list for group joiner, page 2

PeerView usability eksperiment – gennemgangsskema 1

PeerView er et prototype system beregnet til understøttelse af samarbejde mellem grupper. Systemet giver mulighed for at placere et antal dokumenter på en delt arbejdsflade og deltage i grupper med andre som har samme mulighed. Derved præsenteres hver deltager for et samlet billede af gruppens arbejdsområde.

Herunder er opstillet en arbejdsgang bestående af en række opgaver udført v.h.a. PeerView hvor meningen er at opgaverne skal gennemføres i den rækkefølge de optræder.
Når du udfører de enkelte opgaver vil jeg bede dig "tænke højt", altså sige (højt og tydeligt) hvis du får et indfald, har en kommentar eller en kritik på et tidspunkt i forløbet. Hvis du f.eks. synes at et vindue er udformet uheldigt eller at systemet reagerer for langsomt, så sig det. Når alle opgaver er fuldført vil jeg udlevere et kort spørgeskema for at få din vurdering af det system du har arbejdet med.

Du kan til enhver tid åbne PeerViews hjælpesystem ved at vælge det fra "Help" menuen. Der kan du få hjælp til de forskellige bokse der vises undervejs i udførslen af opgaverne.

Arbejdsgang (menigt gruppemedlem)

1. Start en netbrowser og hent PeerView client softwaren fra denne web-adresse: http://home.worldonline.dk/~lars_yde/peerview/
2. Installér PeerView softwaren på den maskine du arbejder ved.
3. Find PeerView i systemets start-menu og start programmet.
4. Åbn boksen "Preferences" ved at vælge det dertil hørende menupunkt eller knap.
5. Aktivér fanebladet "Server" og indtast "Ryger 4" i feltet "Server name". I feltet "Server port" skal der stå 4461 og i feltet "Connection type" skal der stå "Socket".
6. Aktivér fanebladet "Personal" og indtast dit navn samt en kort signatur som du ønsker skal tilføjes de indlæg i "debatten" du kommer til at lave senere. Signatur feltet kan lades blankt hvis du ikke har lyst til at indtaste noget.
7. Tryk OK når du sikret dig at fanebladene har ovenstående indhold.
8. Luk PeerView ved at vælge "Exit" fra menuen eller trykke den tilsvarende knap.
9. Start PeerView igen på samme måde som i trin 3.
10. Åbn "Add documents" boksen ved at vælge den fra PeerViews menu eller ved at klikke den tilsvarende knap på panelet øverst på skærmen.
11. Tilføj et antal filer (mellem 10 og 100) ved at navigere gennem katalogerne på systemets diske, markere filer og flytte dem v.h.a. pileknapperne i boksen. Tryk OK når du har flyttet et tilstrækkeligt antal filer over i "Selected documents" boksen.
12. Find det menupunkt eller knap der åbner oversigten over grupper og vælg det/den.
13. Find gruppen "Test gruppe", markér den og slut dig til gruppen ved at trykke "Join group". Hvis gruppen ikke findes så vent lidt til den er blevet oprettet (skulle ikke tage mere end et par minutter). Når du har sluttet dig til gruppen kan du lukke gruppekataloget ved at trykke "Close".
14. Åbn boksen "Documents in panorama" øverst i hovedvinduet og vælg dokumentet "test_gruppe.txt". Vent til dokumentet er centreret på skærmen og der er dukket et træ op i nederste del af hovedvinduet.

Figure B.3: Task list for group joiner, page 1

15. Dette trin er valgfrit – du behøver ikke udføre det. Markér en portion tekst i dokumentet som du kunne tænke dig at give en kommentar til, højre-klik musen og vælg "Copy text" fra den menu der dukker op (pop-up menu).

16. Markér et element i træet i hovedvinduets nederste venstre hjørne. Klik på den knap der lader dig skrive et indlæg ("Compose new message"). Brug hjælpen fra "Help" menuen hvis du har problemer.

17. Skriv en kommentar til dokumentet og indsæt gerne evt. tekst valgt i trin 15 ved at højreklikke i det område hvor du skriver teksten og vælge "Paste text".

18. Afslut boksen ved at vælge "Submit".

19. Dobbeltklik området udenfor dokumenterne i hovedvinduets midte således at der zoomes ud til et oversigtsniveau hvor du kan overskue alle dokumenterne.

20. Højreklik ovenpå et tilfældigt dokument og vælg "Move". Flyt dokumentet til en ny position. Brug hjælpesystemet hvis du har brug for det. Højreklik igen på dokumentet og sæt et kryds i feltet "Lock" hvis det ikke allerede er gjort.

21. Højreklik på et andet dokument og vælg "Resize". Giv dokumentet en ny størrelse.

22. Højreklik på et tilfældigt dokument og vælg "Arrange all".

23. Vælg dokumentet "test_gruppe.txt" fra "Documents in panorama" boksen. Hvis der er sket en ændring siden du sidst valgte det, så gå videre. Ellers vent til der er sket en ændring.

24. Åbn boksen "Remove documents" og fjern alle dine dokumenter fra systemet. Brug hjælpesystemet hvis du har brug for det.

25. Afslut PeerView ved at vælge "Exit" fra menuen.

Figure B.4: Task list for group joiner, page 2

B.1.2 English version of task list for rank-and-file members

PeerView usability experiment task list 1

PeerView is a prototype system for supporting group collaboration. The system allows its users to place a number of documents in a shared workspace and participate in groups of other users with the same capability. This presents each group participant with a global view of the work undertaken by the group.

The below list contains a number of tasks which must be carried out in the order they appear using the PeerView software. When carrying out the individual tasks, I would like you to "think aloud", that is, say out (loud and clear) what thoughts, comments or critical remarks you might have during the course of events. If, for instance, you think a window has been poorly designed or that the system reacts too slowly, then say so. When all tasks have been completed, I will hand out a brief questionnaire to get your asssessment of the system you will have been working with.

At any time, you can open the PeerView online help system by selecting it from the "Help" menu. This contains online help for all of the different boxes shown as the tasks are carried out.

Task list (rank-and-file member)

1. Launch a webbrowser and download the PeerView client software from the below web address: `http://home.worldonline.dk/ lars_yde/peerview/`.
2. Install the PeerView software on the machine you're working with.
3. Find PeerView in the system "Start" menu and launch the programme.
4. Open the "Preferences" dialog by choosing the corresponding menu item or toolbar button.
5. Activate the "Server" tab page and enter "Ryger 4" in the "Server name" field. In the "Server port" field, you should enter 4461 and you should set the "Connection type" field to "Socket".
6. Activate the "Personal" tab page and enter your name along with a brief signature that you would like to have added to the contributions to ongoing debates that you'll be making later. The signature field may be left empty.
7. Click "OK" when the tab pages are set to the above values.
8. Close PeerView by choosing "Exit" from the drop-down menu or by clicking the corresponding toolbar button.
9. Start PeerView again using the procedure described in step 3.
10. Open the group directory by selecting the "Groups" item from the "Setup" menu.

11. Add a new group by the name "Test gruppe" by clicking the "Create group" button. Then close the box by clicking the "Close" button.

12. Open the "Add Documents" box by choosing it from the "PeerView" drop-down menu or by clicking the corresponding toolbar button.

13. Add the "test_ gruppe.txt" from the "c:\temp" directory by moving it to the "Selected documents" box. Click "OK"

14. Open the "Documents in panorama" box at the top of the main window and choose the "test_gruppe.txt" document. Wait till the document has been centred on screen and a tree [structure] has appeared in the bottom portion of the main window.

15. This step is optional — you don't have to complete it. Select a portion of the text in the [test_ gruppe.txt] document that you would like to comment on. Right-click the mouse and choose "Copy text" from the pop-up menu that then appears.

16. Select an element from the tree in the lower left corner of the main window. Click the button that allows you to write a contribution ("Compose new message"). Use the online help system you have any difficulties.

17. Write a comment on the document and insert the text selected earlier, if any, by right clicking in the editable area and choosing "Paste text".

18. Close the box by choosing "Submit".

19. Double-click the area outside the documents in the center portion of the main window so that it zooms to overview magnification at which you're able to survey all of the documents.

20. Right-click an arbitrary document and select "Move". Move the document to a new position. Use the online help system if you need to. Right-click the document again and enable the "Lock" item if it isn't already enabled.

21. Right-click a different document and choose "Resize". Resize the document.

22. Right-click an arbitrary document and choose "Arrange all".

23. Select the "test_gruppe.txt" document from the "Documents in panorama" box.

24. Change any part of the "text_gruppe.txt" file in the Notepad window that has been opened on your Windows desktop and save the file. Check to see if the change has appeared in the document displayed by PeerView.

25. Open the "Remove documents" box and remove all of your documents from the system. Use the online help system if you need to.

26. Terminate PeerView by choosing "Exit" from the menu.

B.1.3 English version of task list for group creator

PeerView usability experiment task list 1

PeerView is a prototype system for supporting group collaboration. The system allows its users to place a number of documents in a shared workspace and participate in groups of other users with the same capability. This presents each group participant with a global view of the work undertaken by the group.

The below list contains a number of tasks which must be carried out in the order they appear using the PeerView software. When carrying out the individual tasks, I would like you to "think aloud", that is, say out (loud and clear) what thoughts, comments or critical remarks you might have during the course of events. If, for instance, you think a window has been poorly designed or that the system reacts to slowly, then say so. When all tasks have been completed, I will hand out a brief questionnaire to get your asssessment of the system you will have been working with.

At any time, you can open the PeerView online help system by selecting it from the "Help" menu. This contains online help for all of the different boxes shown as the tasks are carried out.

Task list (group creator)

1. Launch a webbrowser and download the PeerView client software from the below web address: `http://home.worldonline.dk/ lars_yde/peerview/`.
2. Install the PeerView software on the machine you're working with.
3. Find PeerView in the system "Start" menu and launch the programme.
4. Open the "Preferences" dialog by choosing the corresponding menu item or toolbar button.
5. Activate the "Server" tab page and enter "Ryger 4" in the "Server name" field. In the "Server port" field, you should enter 4461 and you should set the "Connection type" field to "Socket".
6. Activate the "Personal" tab page and enter your name along with a brief signature that you would like to have added to the contributions to ongoing debates that you'll be making later. The signature field may be left empty.
7. Click "OK" when the tab pages are set to the above values.
8. Close PeerView by choosing "Exit" from the drop-down menu or by clicking the corresponding toolbar button.
9. Start PeerView again using the procedure described in step 3.
10. Open the "Add Documents" box by choosing it from the "PeerView" drop-down menu or by clicking the corresponding toolbar button.

11. Add a number of files (between 10 and 100) by navigating through the directories on the system disk drives, selecting files and moving them using the arrow buttons in the [Add documents] box. Click "OK" when you have moved a sufficient number of files to the "Selected documents" box.

12. Find the menu item or button that opens the group directory and select it.

13. Find the "Test gruppe" group, select it and join it by clicking the "Join group" button. If the group does not exists, then wait a while till it has been created (shouldn't take more than a couple of minutes). Once you've joined the group, you can close the group directory by clicking "Close".

14. Open the "Documents in panorama" box at the top of the main window and choose the "test_ gruppe.txt" document. Wait till the document has been centred on screen and a tree [structure] has appeared in the bottom portion of the main window.

15. This step is optional — you don't have to complete it. Select a portion of the text in the [test_gruppe.txt] document that you would like to comment on. Right-click the mouse and choose "Copy text" from the pop-up menu that then appears.

16. Select an element from the tree in the lower left corner of the main window. Click the button that allows you to write a contribution ("Compose new message"). Use the online help system you have any difficulties.

17. Write a comment on the document and insert the text selected earlier, if any, by right clicking in the editable area and choosing "Paste text".

18. Close the box by choosing "Submit".

19. Double-click the area outside the documents in the center portion of the main window so that it zooms to overview magnification at which you're able to survey all of the documents.

20. Right-click an arbitrary document and select "Move". Move the document to a new position. Use the online help system if you need to. Right-click the document again and enable the "Lock" item if it isn't already enabled.

21. Right-click a different document and choose "Resize". Resize the document.

22. Right-click an arbitrary document and choose "Arrange all".

23. Select the "test_gruppe.txt" document from the "Documents in panorama" box.

24. Change any part of the "text_ ruppe.txt" file in the Notepad window that has been opened on your Windows desktop and save the file. Check to see if the change has appeared in the document displayed by PeerView.

25. Open the "Remove documents" box and remove all of your documents from the system. Use the online help system if you need to.

26. Terminate PeerView by choosing "Exit" from the menu.

B.2 Transscript

B.2.1 Danish version

1. Thomas: Når jeg nu skal downloade
2. Lars: Ja
3. Thomas: Hvad er det så præcist jeg skal trykke på?
4. Lars: Det er klienten du skal downloade
5. Thomas: Klienten ?
6. Lars: Ja
7. Thomas: Det er ikke beskrevet her [i gennemgangsskemaet] ...skal vi hente den med eller uden Java [Java Virtual Machine]
8. Lars: Ja, men det afhænger så af systemet - i det her tilfælde skal i hente den med, så er vi sikker på det kan afvikles.
9. Lars: I skal bare downloade med
10. Rikke: Jeg skal bare trykke her [peger på skærmen]
11. Lars: Ja, tryk bare ...
12. Thomas: Programmet kører direkte [InstallAnywhere installationsprogrammet]
13. Lars: Ja, det kører direkte [over netværksforbindelsen], ja
14. Thomas: Den spørger hvor man skal placere det [installationsprogrammet] ...køre det eller gemme det på...?
15. Lars: øhm, ja du kan bare køre det der ...det tager så lidt tid.

    [ca. 10 minutters pause]
16. Lars: Proceduren her var så ikke så svær at gennemføre. Det der var det største problem var sidt jeg kunne se at indtaste adressen [url'en].
17. Rikke: Det er fordi tastaturet har en anden [opsætning (end den hun kender fra sit arbejde)] ...
18. Lars: [forklarer hvorfor InstallAnywhere er brugt til at lave installationsprogrammet]
19. Paul: Jeg vidste ikke helt hvad for en man skulle vælge af de der [installationsmuligheder] ...det var kun fordi thomas havde sagt det [at der skulle downloades med Java Virtual Machine].
20. Lars: Ja, i virkeligheden skulle folk så helst finde ud af det af sig selv. Jeg står selvfølgelig ikke og kan fortælle dem det når de skal downloade, så det skal helst gøres klarere. Client, det er ikke særlig ...det kræver en teknisk indsigt at se at det er den klient der skal køres. Så det er faktisk en dårlig betegnelse.

21. Lars: Ja, det her Java ... Jeg kunne have valgt at downloade det hele på forhånd, men det lod jeg være med for jeg syntes også i skulle igennem den her del af det.

22. Thomas: Skal jeg taste det der R-y-g-e-r-4 ind ... det er jo tastet ind.

23. Lars: Jamen hvis det er det så er det fordi den husker indstillingerne fra dengang jeg testede [prøveinstallerede] det.

24. Thomas: da jeg trykkede OK, så gik den ud af programmet ... Jeg flyttede nogle filer over [d.v.s mellem listboksene i "Add Documents" boksen], ikke

25. Lars: Ja,

26. Thomas: da jeg så trykkede OK gik den ud af programmet.

27. Lars: BEMÆRKNING. Man må ikke tilføje kataloger i "Add Files" bokse.

28. Rikke: Hvad skal man ...

29. Lars: Du skal først vælge dokumenter heroppe fra ... ja det har du gjort.

30. Lars: Jamen du har ikke forbindelse til serveren endnu. Vi gør lige det vi lukker det ned igen og starter det op. Fordi den har ikke kunnet ... den har mistet forbindelsen. Det sker nogen gange.

31. Rikke: Jeg kan bare starte på [trin] 14 igen.

32. Lars: Ja, den skulle gerne gå igang med at tilføje ting igen nu.

33. Lars: Det er fordi [servernavnet] skal være skrevet med stort [begyndelsesbogstav] ... jeg tror det er derfor ... ja.

34. Lars: Prøv at gøre det samme forløb som du gjorde sidst her. Det var simpelthen fordi den [servernavnet] var med småt der, tror jeg.

35. Thomas: Jeg overførte nogle filer fra [available documents] boksen [til selected documents boksen].

36. Lars: Der var ingen kataloger imellem ?

37. Thomas: Nej

38. Lars: Det var derfor så [at programmet terminerede tidligere] ... så skal du bare køre videre.

39. Thomas: Nå , okay.

40. Rikke: Kan jeg godt markere flere [filer] med CTRL eller skal jeg ...

41. Lars: Nej, det kan du godt, ja

42. Rikke: [she is pointing to the visible documents box and complaining that it fall upwards. The exact words cannot be made out.]

43. Lars: BEMÆRKNING: Ved visse opløsninger så kan visible documents boksen ikke ses fordi den laver orientering vertikalt opadrettet, så det skal fikses. Den skal tvinges til at have orientering vertikalt nedadrettet eller også skal det være brugertilrettet hvordan den orienteres. Et eller andet skal der gøres ihvertfald.

44. Rikke: og så gør jeg ...

45. Lars: Ja, så vælger du og så må du så flytte vinduet op

46. Thomas: Nu har jeg valgt en gruppe - hvordan kan jeg se at jeg har joinet ?

47. Lars: Det [udviklingen af PeerView] er ikke kommet så langt endnu at du kan se de enkelte deltagere. Du kan se et deltagerantal.

48. Lars: JA, men det skal den ogsåPå et tidspunkt skulle det blive sådan at den giver en liste over dem der har deltaget, men det gør den ikke nu.

49. Lars: Hvis du går ud af det her [group directory boksen] vil du kunne se at i [visible documents boksen] der optræder du.

50. Lars: BEMÆRKNING: Der bør laves en liste over deltagerne i stedet for bare en antalsangivelse. Det kunne være med sạn en dropdown-liste, f.eks. i group directory boksen.

51. Rikke: Marker et, øh, det er bare en af de her mapper [i topic tree].

52. Lars: BEMæRKNING: Når man resizer et dokument burde man måske have mulighed for at scrolle viewporten til siderne. Man kan kun resize et dokument i de dimensioner viewporten har når man begynder på det.

53. Lars: I kan resize de forskellige portioner af vinduet ved at hive i de her bjælker de er delt op, altså de her bjælker med den noprede overflade, det er nogen der kan hives i så man kan gøre dem større og mindre hvis man har lyst.

54. Paul: Det her tekstdokument, det er her ikke mere.

55. Lars: Det er fordi du skal centrere det. Nu har du overblik. Du skal centrere det.

56. Lars: [Til Rikke] Nu gør den klar til at bevæge den [indholdet af panoramaet]

57. Rikke: Nå , men jeg tænkte på om jeg så skulle gå ind og ... altså når du har været inde og resize eller er det kun nḏu har flyttet [et dokument].

58. Rikke: ... slutte af, ikke, når du har flyttet det.

59. Lars: Jo, det ...

60. Rikke: så skal jeg til at resize, ikke. Hvad gør jeg så [venstreklikker på panoramaets baggrund]

61. Lars: Det er fordi du venstreklikker. Nu kom du så til at afslutte resize [operationen]. Prøv igen.

62. Lars: Så prøv at flytte derhen hvor du vil have den. Så vent på at den kommer derud. Så klik venstre [museknap].

63. Lars: [Lock] betyder bare at den [dokumentet] låses. Så bliver den stående der.

64. Rikke: På plads ?

65. Lars: Ja

66. Thomas: Prøv at se ... [peger på markering af tekst i tekstdokument]

67. Lars: Prøv at dobbeltklikke ... det er fordi at, øh, det er fordi at den her markering her. Den har een ... den er på et niveau inde i skærmen og teksten er på et andet. Det er fordi den centrerer ikke med.

68. Thomas: Nej, okay.

69. Lars: Prøv at markere igen. Kør den længere ud. stop. tilbage. kør frem igen. stop. slip. prøv at køre lidt ud igen. det er lidt svært [problemer med at styre skaleringen - bevæger sig for hurtigt]. det jeg ville vise var bare at den kun lige nøjagtigt dækker helt når du har den på en helt bestemt. Du får al teksten, Prøv at køre ned igen, så får du hele teksten.

70. Thomas: Nå , okay, så det er bare sådan et synsbedrag.

71. Lars: Synsbedrag, ja. Det er faktisk en meget god ting [at du opdagede det].

72. Lars: [Til Rikke] Den der gruppe.txt [test_gruppe.txt] - kan du huske den ? Så du efter det [om den var ændret på afslutningstidspunktet]

73. Rikke: Jeg kan slet ikke huske hvad det var. Jeg kan bare huske den så ikke ud som dengang jeg startede.

74. Lars: Okay, men, øh, det gør heller ikke så meget. Bare der var sket en ændring.

75. Rikke: Det var der.

76. Lars: Godt, det var det vigtige.

77. Lars: Jeg kan se det samme herovre hos Thomas. [Til Thomas] Når du kommer til det punkt der hedder 23 [i gennemgangsskemaet] så lad lige mig se hvad der er sket.

78. Lars: BEMÆRKNING: Når man markerer en portion tekst, highlighter den inden man vælger copy text, så er highlightingen ikke på samme skalering som teksten. Det giver sådan et sjovt synsbedrag. Man får godt nok kopieret den rigtige mængde tekst, men det syner ikke så dan for brugeren. Det er ikke så heldigt.

79. Lars: [Til thomas] Det jeg skulle se var bare om de ændringer der var blevet lavet de også var blevet sendt rundt [Paul havde forladt gruppen inden Thomas kunne inspicere det delte dokument "test_gruppe.txt"].

B.2.2 English version

1. Thomas: When I'm about to download ...
2. Lars: Yes ?
3. Thomas: What is it then, specifically, that I should click ?
4. Lars: You're supposed to download the client.
5. Thomas: The client ?
6. Lars: Yes.
7. Thomas: That's not described in here [in the task list] ...should we download it with or without Java [The Java Virtual Machine].
8. Lars: Well, that then depends on the system - in this case you should get it with [the Java VM]. That way we're sure it can execute.
9. Lars: You should just download including [the Java VM].
10. Rikke: I just click here [points to the display].
11. Lars: Yes, just click ...
12. Thomas: The programme executes directly [The InstallAnywhere installation programme].
13. Lars: Yes, it executes directly [over the network connection], yes.
14. Thomas: It asks me where to put it [the installation programme] ...run it or save it to ...?
15. Lars: Ehm, yes, you can just run it ...that'll then take some time.

    [approximately 10 minutes intermission]
16. Lars: This [installation] procedure wasn't that difficult to complete. What appeared to me to be the biggest problem was entering the URL.
17. Rikke: That's because the keyboard is configured differently [to her workplace configuration].
18. Lars: [explains the reason for using InstallAnywhere for the installation programme].
19. Paul: I didn't quite know which one of those [download options] to select ...it was only because Thomas said [that the download should include the Java VM].
20. Lars: Yes, in real life people should then find that out by themselves. I can't be there to tell them what to download, of course, so that should be made clearer. Client, that's not very ...it takes technical insight to see that that is the executable client programme. So that is actually a bad name for it.
21. Lars: Well, this entire Java thing ...I could have downloaded it all in advance, but I didn't because I thought you should go through that part as well.

22. Thomas: Should I enter the R-y-g-e-r-4 bit [the server name to enter in the preferences dialog box] - that's already been entered.

23. Lars: Well, that must be because it remembers the settings from when I [test installed] the programme.

24. Thomas: When I pressed OK, it exited the programme ... I transferred some files [from one list box in the "Add Documents" dialog to another], right ?

25. Lars: Yes.

26. Thomas: Then when I pressed OK it exited the programme.

27. Lars: COMMENT: Users should not be able to add directories to the "Selected documents" list box in the "Add Documents" dialog box.

28. Rikke: What are you supposed to ...

29. Lars: You must first select documents from up here [points to the "Visible documents" box] ... Yes, you already have.

30. Lars: But there's no connection to the server yet. We'll just close it [the application] and restart it. Since it hasn't been able to ... it's lost the connection. That sometimes happens.

31. Rikke: I can just start over from [step] 14 again.

32. Lars: Yes, it [the client application] should be adding [documents] to the panorama now.

33. Lars: It's because [the server name] must be written with a capital [first letter] ... I think that's the reason why ... yes.

34. Lars: Try going through the same procedure as you did the last time at this point. It was simply because it [the server name] had a lower case first letter, I think.

35. Thomas: I transferred some files from [the available documents] list box [to the selected documents list box].

36. Lars: There were no directories among [the selected items].

37. Thomas: No.

38. Lars: That was the reason [why the application had exited earlier] ... just carry on, then.

39. Thomas: Oh, okay.

40. Rikke: Can I select multiple files using CTRL or must I ...

41. Lars: No, you can do that ... yes.

42. Rikke: [she is pointing to the visible documents box and complaining that it falls upwards. The exact words can't be made out.]

43. Lars: COMMENT: At some resolutions the visible documents box becomes invisible because its orientation is upwards, so that should be fixed. It should be forced to drop down or its orientation should be user customizable. Something has to be done.

44. Rikke: And then I ...

45. Lars: Yes, then you select [an item from the visible documents box] and then you have to move the window up [she has moved the window down so that she can select an item the visible documents box].

46. Thomas: I've selected a group now - how can I tell that I've joined ?

47. Lars: That [the development of PeerView] hasn't come so far that you can see the individual members [of a group]. You can see the number of participants.

48. Thomas: [suggests it would be useful to be able to tell who are in a group].

49. Lars: Yes, but it [PeerView] is supposed to. At some point it is supposed to be able to produce a list of those who have participated, but it doesn't do that at the moment.

50. Lars: If you end this [the group directory dialog box] you can see that you appear in [the visible documents box].

51. Lars: COMMENT: There ought to be a list of participants rather than just their number. This could be done with a drop-down list of sorts, for instance in the group directory box.

52. Rikke: Select a, ehm, its just one of these folders [in the topic tree].

53. Lars: COMMENT: [After observing Rikke having problems with a resize operation]. When resizing a document, you should perhaps be able to scroll the viewport to either side. You can only resize a documents to fit the dimensions of the viewport at the time when you begin [the resize operation].

54. Lars: You can resize the different parts of the [PeerView client main] window by pulling the [divider] bars that separate them, that is, the bars with the granulated surface, they can be pulled so you can make them bigger and smaller if you like.

55. Paul: This text document [test_ gruppe.txt], its not here anymore.

56. Lars: That's because you have to centre it. Now you're at overview [position]. You have to centre it.

57. Lars: [Speaking to Rikke]. Its [PeerView] preparing to move them [the panorama contents].

58. Rikke: Well, but I was wondering whether I should go in and ...I mean, when you've resized something or is it only when you've moved [a document]. [She is in doubt about how to carry out the resize and move operations].

59. Rikke: ...end [the operation], right ...when you've moved it ?

60. Lars: Yes, that ...

61. Rikke: now I'm about to resize, right ? What do I do next [left clicking on the panorama background].

62. Lars: That's because you're left clicking. You accidentally ended the resize [operation]. Try again.

63. Lars: Then try and move to where you want it. Then wait for it to move there [the resize to take effect]. Then click left [mouse button].

64. Lars: [Lock] simply means that it [the document] is locked. It will then stay there [at a fixed position].

65. Rikke: In place ?

66. Lars: Yes.

67. Thomas: Take a look ... [points to highlighted text in a text document].

68. Lars: Try double-clicking ... that's because, ehm, that's because the highlighting here ... it has a ... its at one level of the display and the text is at another. That's because it [the highlighting] doesn't scale [with the text].

69. Thomas: Oh, okay.

70. Lars: Try highlighting it again. Move it further out. Stop. Back up. Move forward again. Stop. Release. Try moving a bit forward again. That's a bit difficult [Thomas has difficulty controlling the scaling operation - the panorama contents scale too fast]. What I wanted to show was just that it [the highlighting] only covers [the text] completely when it is at a very specific [level of magnification]. You'll get all the text, try scrolling down again, you'll get all of the [highlighted] text.

71. Thomas: Oh, like that, so its just a kind of optical illusion.

72. Lars: Optical illusion, yes. Its actually quite a good thing [that you discovered that].

73. Lars: [Speaking to Rikke]. That gruppe.txt [the file test_ gruppe.txt] - do you remember that ? Did you check [if it had been changed when you exited PeerView] ?

74. Rikke: I can't at all remember what it [looked like]. I just remember that it didn't look the same as when I started.

75. Lars: Okay, but ehm, that's not really important. As long as there was a change.

76. Rikke: There was a change.

77. Lars: Good, that's what's important.

78. Lars: I should be able to see the same thing over here by Thomas. [Speaking to Thomas]. When you reach item 23 [in the task list], then let me see what has happened.

79. Lars: COMMENT: When you select some text — highlight it before copying it — the highlighting is not at the same level of magnification as the text. That produces a peculiar optical illusion. You copy the intended amount of text, all right, but it doesn't appear that way to the user. That's not very sensible.

80. Lars: [Speaking to Thomas]. What I wanted to see was just whether the changes that had been made had been distributed [Paul had left the group before Thomas could inspect the shared document "test_ gruppe.txt" that Paul had changed, but Rikke had already confirmed that the change had been distributed and applied. Also, I had tested this feature independently on other occasions and saw no reason to repeat the latter steps of the experiment.]

B.3 Questionnaires

B.3.1 Danish version

The below are scanned reproductions of the original questionnaires as they were returned by the experiment participants. An English translation can be found in the next Section.

PeerView spørgeskema

Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

	Helt enig	Enig	Neutral	Uenig	Helt uenig
Jeg fandt det let at installere PeerView.	X				
Jeg fandt grænsefladen let at arbejde med.	X				
Jeg synes systemet reagerede for langsomt på mine kommandoer.				X	
Jeg synes udformningen af grænsefladen (farver/formgivning) hjalp mig i mit arbejde.		X			
Jeg synes PeerView var behageligt at arbejde med.		X			
Jeg havde problemer med at forstå PeerViews meddelelser til mig.				X	
Jeg havde problemer med at bruge PeerViews funktioner.				X	
Jeg synes hjælpesystemet var tilfredsstillende.			X		
Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrækkelige.		X			
Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.					X
Jeg synes det var let at tilføje beskeder/emner til træet.		X			

Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

	I høj grad	**I nogen grad**	**Ingen af delene**	**I nogen grad**	**I høj grad**	
Nyttigt		X				Unyttigt
Let tilgængeligt		X				Svært at bruge
Pænt	X					Grimt
Sjovt at bruge			X			Kedeligt at bruge
Hurtigt	X					Langsomt

Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitiøst system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder:

Jeg synes at der skal oplyses inden man påbegynder at bruge systemet, hvad det kan bruges til, og hvilke situationer det vil gøre sig særdeles egnet til.
På den måde vil der blive skabt en rød tråd for nye bruger, som måske ikke har den store indsigt i, hvad det hele går ud på.
Dog skal det siges, at jeg godt er klar, at programmet henvender sig til en bestemt slags bruger, hvilke specifikt ved hvad de skal bruge programmet til.

PeerView spørgeskema

Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

	Helt enig	Enig	Neutral	Uenig	Helt uenig
Jeg fandt det let at installere PeerView.		X			
Jeg fandt grænsefladen let at arbejde med.		X			
Jeg synes systemet reagerede for langsomt på mine kommandoer.				X	
Jeg synes udformningen af grænsefladen (farver/formgivning) hjalp mig i mit arbejde.			X		
Jeg synes PeerView var behageligt at arbejde med.		X			
Jeg havde problemer med at forstå PeerViews meddelelser til mig.			X		
Jeg havde problemer med at bruge PeerViews funktioner.				X GANSKE LIDT	
Jeg synes hjælpesystemet var tilfredsstillende.			X BRUGTE IKKE HJÆLPE FUNKTIONEN		
Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrækkelige.	X				
Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.					X
Jeg synes det var let at tilføje beskeder/emner til træet.	X				

Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

	I høj grad	**I nogen grad**	**Ingen af delene**	**I nogen grad**	**I høj grad**	
Nyttigt			X			Unyttigt
Let tilgængeligt		X				Svært at bruge
Pænt	X					Grimt
Sjovt at bruge	X					Kedeligt at bruge
Hurtigt		X				Langsomt

VED IKKE HVAD JEG SOM PERSON SKULLE BRUGE PROGRAMMET TIL???

Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitiøst system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder:

- DE PROBLEMER JEG HAVDE UNDERVEJS SKYLDTES TIL OELS, AT "TINGENE" STOD PÅ ENGELSK (BETYDNINGEN "FORSVANDT" FOR MIG).
- JEG SKULLE BRUGE MUSEN MEGET – GENVEJSTASTERNE "FORSVANDT"

PeerView spørgeskema

Anvendelighedsspørgsmål

Her bedes du angive i hvor høj grad du er enig i nedenstående udsagn.

	Helt enig	Enig	Neutral	Uenig	Helt uenig
Jeg fandt det let at installere PeerView.		X			
Jeg fandt grænsefladen let at arbejde med.			X		
Jeg synes systemet reagerede for langsomt på mine kommandoer.					X
Jeg synes udformningen af grænsefladen (farver/formgivning) hjalp mig i mit arbejde.			X		
Jeg synes PeerView var behageligt at arbejde med.	X				
Jeg havde problemer med at forstå PeerViews meddelelser til mig.			X		
Jeg havde problemer med at bruge PeerViews funktioner.			X		
☆ Jeg synes hjælpesystemet var tilfredsstillende.					
Jeg synes mulighederne for at flytte, forme og ordne dokumenter var tilstrækkelige.	X				
Jeg synes træstrukturen nederst i hovedvinduet var svær at bruge.		X			
Jeg synes det var let at tilføje beskeder/emner til træet.	X				

Vurderingsspørgsmål

Her bedes du angive i hvor høj grad du synes nedenstående udtryk betegner PeerView.

	I høj grad	I nogen grad	Ingen af delene	I nogen grad	I høj grad	
Nyttigt	X					Unyttigt
Let tilgængeligt		X				Svært at bruge
Pænt		X				Grimt
Sjovt at bruge	X					Kedeligt at bruge
Hurtigt	X					Langsomt

☆ Brugte den ikke

Figure B.9: Questionnaire 3, page 1

Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreudvikles til et mere ambitiøst system. Hvis du har forslag, kommentarer eller kritik du mener vil kunne bidrage til dette produkt bedes du skrive dem herunder:

- Måske kan man installer prg. i windows og brug princippet til at udveksle problemløsninger etc.

- Jeg har indtrykket af at man hurtig vil blive "DUS" med programmet, og vil glad for det i det rette miljø.

- Jeg syntes også at de engelske ord er ret tekniske for en "normal" bruger.

-

Thomas

B.3.2 English version of PeerView questionnaire for first participant

Usability questions

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.

	Agree entirely	Agree	Neutral	Disagree	Disagree entirely
I found it easy to install PeerView	X				
I found the interface easy to work with	X				
I thought the system reacted to slowly to my commands			X		
I thought the interface design (colours/layout) made my work easier		X			
I found PeerView pleasant to work with		X			
I had difficulty understanding PeerView messages				X	
I found the online help system satisfactory			X		
I thought the facilities for moving, resizing and arranging document satisfactory		X			
I found the tree structure at the bottom of the main window easy to use					X
I found it easy to add messages/topics to the tree	X				

Evaluation questions

Please indicate in the below table the degree to which you think the phrases in the left and rightmost columns apply to PeerView.

	Very	Somewhat	Neither	Somewhat	Very	
Useful		X				Useless
Accessible		X				Inaccessible
Attractive	X					Unattractive
Fun to use			X			Dull to use
Quick to respond	X					Slow to respond

Comments

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

I think the user should be made aware of what the programme is supposed to be used for and the types of situation it could used in before it is put to use. That way a novice user who may not have great insight into the purpose of it all will have a guideline. I do realize, though, that the programme is aimed at a particular type of user who will be well aware of what the programme is for.

B.3.3 English version of PeerView questionnaire for second participant

Usability questions

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.

	Agree entirely	Agree	Neutral	Disagree	Disagree entirely
I found it easy to install PeerView		X			
I found the interface easy to work with		X			
I thought the system reacted to slowly to my commands				X	
I thought the interface design (colours/layout) made my work easier			X		
I found PeerView pleasant to work with		X			
I had difficulty understanding PeerView messages			X		
I found the online help system satisfactory			X		
I thought the facilities for moving, resizing and arranging document satisfactory	X				
I found the tree structure at the bottom of the main window easy to use					X
I found it easy to add messages/topics to the tree	X				

Evaluation questions

Please indicate in the below table the degree to which you think the phrases in the left and rightmost columns apply to PeerView.

	Very	Somewhat	Neither	Somewhat	Very	
Useful			X			Useless
Accessible		X				Inaccessible
Attractive	X					Unattractive
Fun to use	X					Dull to use
Quick to respond	X					Slow to respond

Comments

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

- The problems I had were in part due to "things" being in English (the meaning "slipped" from me).
- I had to use the mouse a lot — the shortcut keys "disappeared".

B.3.4 English version of PeerView questionnaire for third participant

Usability questions

Please indicate in the below table the degree to which you agree with the statements in the leftmost column.

	Agree entirely	Agree	Neutral	Disagree	Disagree entirely
I found it easy to install PeerView		X			
I found the interface easy to work with			X		
I thought the system reacted to slowly to my commands					X
I thought the interface design (colours/layout) made my work easier			X		
I found PeerView pleasant to work with	X				
I had difficulty understanding PeerView messages			X		
I found the online help system satisfactory					
I thought the facilities for moving, resizing and arranging document satisfactory	X				
I found the tree structure at the bottom of the main window easy to use		X			
I found it easy to add messages/topics to the tree	X				

Evaluation questions

Please indicate in the below table the degree to which you think the phrases in the left and rightmost columns apply to PeerView.

	Very	Somewhat	Neither	Somewhat	Very	
Useful	X					Useless
Accessible		X				Inaccessible
Attractive		X				Unattractive
Fun to use	X					Dull to use
Quick to respond	X					Slow to respond

Comments

PeerView is a prototype and the plan is that at some later stage it should be developed into a more ambitious system. If you have suggestions, comments or critical remarks you believe could contribute to this system, then please write them below:

- Maybe you could install the programme under Windows and use the PeerView functionality to exchange problem solutions etc.

- I have the impression that you would quickly "become acquainted" with the programme and would grow to like it in the proper setting.
- I also think that the English words will seem fairly technical to the "ordinary" user.

Appendix C

Source code

C.1 Introduction

This Appendix contains the source code for PeerView. It is divided into three major Sections corresponding to the major packages of the PeerView application and is prefaced by a table of contents to help the reader find his or her way. The source code references packages that are not listed here, notably the Jazz and JSDT libraries, but all listings can be understood without knowledge of the source code for those packages. If, however, you would like to inspect said code or learn more of these libraries, the source code for Jazz is available at [34] and that for the JSDT at [**?**]. The PeerView source code is available at [54] as is the associated Javadoc documentation and the online help system listed in Appendix D.

C.2 List of listings

Listings

C.1 AddDocumentsAccessory.java 114
C.2 AddFiles.java 119
C.3 ClientApplication.java 122
C.4 ClientInfo.java 166
C.5 ClientManager.java 171
C.6 ClientConstants.java 200
C.7 ClientPanorama.java 226
C.8 ClientPanoramaLayoutManager.java 252
C.9 ComposeNewMessage.java 255
C.10 DiscussionFrame.java 266
C.11 GridLayout.java 288
C.12 GridPanoramaLayoutManager.java 290
C.13 GroupDirectory.java 302
C.14 MessagePropertiesBox.java 318
C.15 MessageWindow.java 323
C.16 PanoramaLayout.java 329
C.17 PeerViewClient.java 331
C.18 PreferencesDialog.java 332
C.19 PZCamera.java 351
C.20 PZCameraListener.java 352
C.21 PZoomHandler.java 352
C.22 PZPanEventHandler.java 353
C.23 PZSwing.java 354
C.24 PZText.java 356
C.25 RemoveFiles.java 357
C.26 Validator.java 367
C.27 GroupManagementServer.java 368
C.28 PeerViewServer.java 369

C.29 ServerApplication.java 370
C.30 ServerConstants.java 384
C.31 ServerInfo.java 388
C.32 ServerManager.java 392
C.33 ServerSetup.java 410
C.34 AboutDialog.java 418
C.35 CompressedDocumentPackage.java 422
C.36 ConfirmationBox.java 423
C.37 DataPackage.java 429
C.38 DeleteGroup.java 431
C.39 DisplayGroup.java 433
C.40 DocPackage.java 441
C.41 EditGroup.java 443
C.42 ErrorBox.java 445
C.43 HelpDialog.java 450
C.44 Message.java 455
C.45 MessageBox.java 458
C.46 NewGroup.java 463
C.47 Pair.java 466
C.48 SharedConstants.java 467
C.49 WorkGroup.java 476

Listing C.1: AddDocumentsAccessory.java

```
package clientapp;

/**
 * This class defines the right hand side interface components of the
 "Add documents" dialog box, i.e. the "Selected documents" list box
 and the two arrow buttons used for transferring files into and out
 from it.
 * Creation date: (15-06-00 15:29:40)
 * @author:
 */

import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;

public class AddDocumentsAccessory extends JPanel {
   private JButton ivjAddButton = null;
   private JPanel ivjButtonPanel = null;
   private JList ivjDocumentList = null;
   private JScrollPane ivjJScrollPane = null;
   private JButton ivjRemoveButton = null;
   private javax.swing.DefaultListModel selectedDocuments;
   private JLabel ivjJLabel = null;
```

```
/**
 * AddDocumentsAccessory constructor comment.
 */
public AddDocumentsAccessory() {
   super();
   initialize();
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param layout java.awt.LayoutManager
 */
public AddDocumentsAccessory(java.awt.LayoutManager layout) {
   super(layout);
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param layout java.awt.LayoutManager
 * @param isDoubleBuffered boolean
 */
public AddDocumentsAccessory(java.awt.LayoutManager layout, boolean isDoubleBuffered) {
   super(layout, isDoubleBuffered);
}
/**
 * AddDocumentsAccessory constructor comment.
 * @param isDoubleBuffered boolean
 */
public AddDocumentsAccessory(boolean isDoubleBuffered) {
   super(isDoubleBuffered);
}
/**
 * Return the AddButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getAddButton() {
   if (ivjAddButton == null) {
      try {
         ivjAddButton = new javax.swing.JButton();
         ivjAddButton.setName("AddButton");
         ivjAddButton.setToolTipText("Add␣to␣list");
         ivjAddButton.setMnemonic('a');
         ivjAddButton.setText("");
         ivjAddButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
             navigation/Forward24.gif")));
         ivjAddButton.setBorderPainted(false);
         ivjAddButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAddButton;
}
/**
 * Return the ButtonPanel property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
   if (ivjButtonPanel == null) {
      try {
         ivjButtonPanel = new javax.swing.JPanel();
         ivjButtonPanel.setName("ButtonPanel");
```

```
            ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
            ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
            ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));

            java.awt.GridBagConstraints constraintsAddButton = new java.awt.GridBagConstraints();
            constraintsAddButton.gridx = 0; constraintsAddButton.gridy = 0;
            getButtonPanel().add(getAddButton(), constraintsAddButton);

            java.awt.GridBagConstraints constraintsRemoveButton = new java.awt.GridBagConstraints();
            constraintsRemoveButton.gridx = 0; constraintsRemoveButton.gridy = 1;
            getButtonPanel().add(getRemoveButton(), constraintsRemoveButton);
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjButtonPanel;
   }
   /**
    * Return the DocumentList property value.
    * @return javax.swing.JList
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   public javax.swing.JList getDocumentList() {
      if (ivjDocumentList == null) {
         try {
            ivjDocumentList = new javax.swing.JList();
            ivjDocumentList.setName("DocumentList");
            ivjDocumentList.setToolTipText("Documents␣displayed␣in␣panorama");
            ivjDocumentList.setBounds(0, 0, 112, 253);
            // user code begin {1}
            selectedDocuments = new DefaultListModel();
            ivjDocumentList.setModel(selectedDocuments);
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjDocumentList;
   }
   /**
    * Return the JLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getJLabel() {
      if (ivjJLabel == null) {
         try {
            ivjJLabel = new javax.swing.JLabel();
            ivjJLabel.setName("JLabel");
            ivjJLabel.setText("␣␣Selected␣documents:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJLabel;
   }
```

```
/**
 * Return the JScrollPane property value.
 * @return javax.swing.JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JScrollPane getJScrollPane() {
   if (ivjJScrollPane == null) {
      try {
         ivjJScrollPane = new javax.swing.JScrollPane();
         ivjJScrollPane.setName("JScrollPane");
         getJScrollPane().setViewportView(getDocumentList());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjJScrollPane;
}
/**
 * Return the RemoveButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getRemoveButton() {
   if (ivjRemoveButton == null) {
      try {
         ivjRemoveButton = new javax.swing.JButton();
         ivjRemoveButton.setName("RemoveButton");
         ivjRemoveButton.setToolTipText("Remove␣from␣list");
         ivjRemoveButton.setMnemonic('r');
         ivjRemoveButton.setText("");
         ivjRemoveButton.setMaximumSize(new java.awt.Dimension(29, 35));
         ivjRemoveButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics
              /navigation/Back24.gif")));
         ivjRemoveButton.setBorderPainted(false);
         ivjRemoveButton.setPreferredSize(new java.awt.Dimension(29, 35));
         ivjRemoveButton.setMinimumSize(new java.awt.Dimension(29, 35));
         ivjRemoveButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjRemoveButton;
}
/**
 * Insert the method's description here.
 * Creation date: (15-06-00 16:07:02)
 * @return javax.swing.DefaultListModel
 */
public javax.swing.DefaultListModel getSelectedDocuments() {
   return selectedDocuments;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
```

```
222     // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
224  }
     /**
226   * Initialize the class.
      */
228  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
230     try {
           // user code begin {1}
232        // user code end
           setName("AddDocumentsAccessory");
234        setLayout(new java.awt.GridBagLayout());
           setSize(329, 319);
236
           java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
238        constraintsButtonPanel.gridx = 0; constraintsButtonPanel.gridy = 1;
           constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
240        constraintsButtonPanel.anchor = java.awt.GridBagConstraints.WEST;
           constraintsButtonPanel.weightx = 1.0;
242        constraintsButtonPanel.weighty = 1.0;
           add(getButtonPanel(), constraintsButtonPanel);
244
           java.awt.GridBagConstraints constraintsJScrollPane = new java.awt.GridBagConstraints();
246        constraintsJScrollPane.gridx = 0; constraintsJScrollPane.gridy = 1;
           constraintsJScrollPane.fill = java.awt.GridBagConstraints.BOTH;
248        constraintsJScrollPane.anchor = java.awt.GridBagConstraints.EAST;
           constraintsJScrollPane.weightx = 1.0;
250        constraintsJScrollPane.weighty = 1.0;
           constraintsJScrollPane.insets = new java.awt.Insets(4, 33, 4, 4);
252        add(getJScrollPane(), constraintsJScrollPane);

254        java.awt.GridBagConstraints constraintsJLabel = new java.awt.GridBagConstraints();
           constraintsJLabel.gridx = 0; constraintsJLabel.gridy = 0;
256        constraintsJLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
           constraintsJLabel.anchor = java.awt.GridBagConstraints.NORTH;
258        constraintsJLabel.insets = new java.awt.Insets(6, 20, 6, 6);
           add(getJLabel(), constraintsJLabel);
260     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
262     }
        // user code begin {2}
264     // user code end
     }
266  /**
      * main entrypoint - starts the part when it is run as an application
268   * @param args java.lang.String[]
      */
270  public static void main(java.lang.String[] args) {
        try {
272        javax.swing.JFrame frame = new javax.swing.JFrame();
           AddDocumentsAccessory aAddDocumentsAccessory;
274        aAddDocumentsAccessory = new AddDocumentsAccessory();
           frame.setContentPane(aAddDocumentsAccessory);
276        frame.setSize(aAddDocumentsAccessory.getSize());
           frame.addWindowListener(new java.awt.event.WindowAdapter() {
278           public void windowClosing(java.awt.event.WindowEvent e) {
                 System.exit(0);
280           };
           });
282        frame.setVisible(true);
        } catch (Throwable exception) {
284        System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JPanel");
           exception.printStackTrace(System.out);
286     }
     }
```

```
/**
 * Set the selectedDocuments property
 * Creation date: (15-06-00 16:07:02)
 * @param newSelectedDocuments javax.swing.DefaultListModel
 */
public void setSelectedDocuments(javax.swing.DefaultListModel newSelectedDocuments) {
   selectedDocuments = newSelectedDocuments;
}
}
```

Listing C.2: AddFiles.java

```
package clientapp;

/**
 * The document used for adding files/documents to the panorama.
 * Creation date: (23-05-00 08:55:51)
 * @author:
 */
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.filechooser.FileFilter;
import java.io.File.*;

public class AddFiles extends JDialog {

   private JPanel ivjJDialogContentPane = null;
   private PeerViewFileChooser ivjFileChooser = null;
   private ClientApplication clientApplication;
/**
 * Constructor
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public AddFiles() {
   super();
   initialize();
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Dialog
 */
public AddFiles(java.awt.Dialog owner) {
   super(owner);
   initialize();
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public AddFiles(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public AddFiles(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Dialog
```

```
 * @param modal boolean
 */
public AddFiles(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Frame
 */
public AddFiles(java.awt.Frame owner) {
   super(owner);
   initialize();
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public AddFiles(java.awt.Frame owner, String title) {
   super(owner, title);
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public AddFiles(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * AddFiles constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public AddFiles(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * Override the JDialog dispose method to allow the current file filter selection to be saved before
     disposing of the box.
 * Creation date: (25-02-01 20:23:39)
 */
public void dispose()
{
   getFileChooser().setPreviousFileFilterSelection( getFileChooser().getFileFilter() );
   super.dispose();
}
/**
 * Insert the method's description here.
 * Creation date: (16-06-00 08:34:47)
 * @return clientapp.ClientApplication
 */
public ClientApplication getClientApplication() {
   return clientApplication;
}
/**
 * Return the FileChooser property value.
 * @return clientapp.PeerViewFileChooser
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private PeerViewFileChooser getFileChooser() {
   if (ivjFileChooser == null) {
      try {
         ivjFileChooser = new clientapp.PeerViewFileChooser();
         ivjFileChooser.setName("FileChooser");
```

```
            ivjFileChooser.setMaximumSize(new java.awt.Dimension(400, 150));
            java.io.File ivjLocal41selectedFiles [] = {};
            ivjFileChooser.setSelectedFiles(ivjLocal41selectedFiles);
            ivjFileChooser.setPreferredSize(new java.awt.Dimension(400, 150));
            ivjFileChooser.setMultiSelectionEnabled(true);
            ivjFileChooser.setMinimumSize(new java.awt.Dimension(300, 150));
            // user code begin {1}

            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjFileChooser;
   }
   /**
    * Return the JDialogContentPane property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJDialogContentPane() {
      if (ivjJDialogContentPane == null) {
         try {
            ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setPreferredSize(new java.awt.Dimension(550, 350));
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            ivjJDialogContentPane.setMinimumSize(new java.awt.Dimension(400, 250));
            getJDialogContentPane().add(getFileChooser(), "Center");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJDialogContentPane;
   }
   /**
    * Called whenever the part throws an exception.
    * @param exception java.lang.Throwable
    */
   private void handleException(java.lang.Throwable exception) {

      /* Uncomment the following lines to print uncaught exceptions to stdout */
      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
      // exception.printStackTrace(System.out);
   }
   /**
    * Initialize the class.
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   public void initialize() {
      try {
         // user code begin {1}
         // user code end
         setName("AddFiles");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         setResizable(true);
         setSize(550, 318);
         setModal(true);
         setTitle("Add␣Documents");
         setContentPane(getJDialogContentPane());
```

```
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      // user code begin {2}
      ivjFileChooser.setAddFilesDialog(this);
      // default positioning puts the dialog at center of screen. The method used is a slight hack, but
           works fine.
      setLocationRelativeTo( getOwner() );
      // user code end
   }
   /**
    * main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
    */
   public static void main(java.lang.String[] args) {
      try {
         AddFiles aAddFiles;
         aAddFiles = new AddFiles((JFrame)null);
         aAddFiles.setModal(true);
         aAddFiles.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
            };
         });
         aAddFiles.setVisible(true);
      } catch (Throwable exception) {
         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
      }
   }
   /**
    * Insert the method's description here.
    * Creation date: (16-06-00 08:34:47)
    * @param newClientApplication clientapp.ClientApplication
    */
   public void setClientApplication(ClientApplication newClientApplication) {
      clientApplication = newClientApplication;
   }
   /**
    * Overrides the JDialog show method to ensure that init is called before the box is displayed.
    * Creation date: (03-08-00 01:22:58)
    */
   public void show()
   {
      getFileChooser().init();
      super.show();
   }
   /**
    * Add the files selected by the user to the panorama and dispose of the dialog box.
    * Creation date: (12-06-00 07:47:56)
    */
   public void submitAndClose()
   {
      Object selectedObjects[] = getFileChooser().getFileChooserAccessory().getSelectedDocuments().toArray
           ();
      File selectedFiles[] = new File[selectedObjects.length];
      for (int i = 0; i < selectedFiles.length; i++)
      {  selectedFiles[i] = (File)selectedObjects[i]; }
      clientApplication.getClientManager().addDocuments(selectedFiles);
      clientApplication.repaint();
      dispose();
   }
   }
```

Listing C.3: ClientApplication.java

```
package clientapp;

/**
 * The main class of the PeerView client application.
 * Creation date: (14-06-00 15:36:49)
 * @author:
 */
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.Properties;

import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import javax.swing.event.*;
import javax.swing.JProgressBar.*;
import javax.swing.Timer.*;

import edu.umd.cs.jazz.*;
import edu.umd.cs.jazz.util.*;
import edu.umd.cs.jazz.event.*;
import edu.umd.cs.jazz.component.*;
import javax.swing.tree.*;

import com.sun.media.jsdt.*;
import peerviewmisc.*;
import javax.help.*;

/** The client application main class */
public class ClientApplication extends JFrame
{
   /** Listener class for activating JavaHelp system
   */
   public class HelpActionListener implements ActionListener
   {
      HelpBroker helpBroker;

      public HelpActionListener( HelpBroker hb )
      {
         helpBroker = hb;
      }
      /** Event handler. Makes the online help window visible and centers it on screen */
      public void actionPerformed( ActionEvent ae )
      {
         helpBroker.setDisplayed( true );
         // calculate placement of help window
         Point winLocation = getLocationOnScreen();
         Dimension winSize = getSize();
         int helpX = (int) ( winLocation.getX() + ( winSize.getWidth() - helpBroker.getSize().getWidth()
               )/ 2 );
         int helpY = (int) ( winLocation.getY() + ( winSize.getHeight() - helpBroker.getSize().getHeight
              () )/ 2 );
         helpBroker.setLocation( new Point( helpX, helpY ));
      }
   };

   /** Objects of this class can be used as caption item in the visible documents box in the client
        application toolbar */
   public static class CaptionItem
   {
      String caption;
      public String getCaption()
      {
```

```
64          return caption;
         }
66       public CaptionItem( String captionArg )
         {
68          caption = captionArg;
         }
70       public String toString()
         {
72          return caption;
         }
74    };
      /** Objects of this class can be used as author items in the visible documents box in the client
           application toolbar */
76    public static class AuthorItem
      {
78       String authorName;
         public String getAuthorName()
80       {
            return authorName;
82       }
         public AuthorItem( String authorNameArg )
84       {
            authorName = authorNameArg;
86       }
         public String toString()
88       {
            return getAuthorName() + ":";
90       }
      };
92    /** Objects of this class can be used as document entries in the visible documents box in the client
           application toolbar */
      public static class DocumentItem
94    {
         String documentID;
96       String documentName;

98       public String getDocumentID()
         {
100         return documentID;
         }
102      public String getDocumentName()
         {
104         return documentName;
         }
106      public DocumentItem( String documentIDArg, String documentNameArg )
         {
108         documentID = documentIDArg;
            documentName = documentNameArg;
110      }
         public String toString()
112      {
            return "\t" + getDocumentName();
114      }
      };
116   /** Event handler for the visible documents box which is the dropdown box on the client application
           toolbar where the
      * names of the documents displayed in the panorama appear
118   */
      public class VisibleDocumentsBoxItemListener implements ItemListener
120   {
         int previousSelectedIndex = 0;
122
         public void itemStateChanged( ItemEvent ie )
124      {
            if ( ( isItemListenerActive() == true )&&
```

```
126             ( (ie.getStateChange() == ItemEvent.SELECTED) || ( getDocumentBox().getSelectedIndex()
                    == 0 )))
          {
128          getDocumentBox().hidePopup();
             if ( ie.getItem() instanceof DocumentItem )
130          {
                getPanoramaPanel().centerDocument( ( (DocumentItem) ie.getItem() ).getDocumentID() );
132          }
             else if ( ie.getItem() instanceof AuthorItem )
134          {
                int currentSelectedIndex = getDocumentBox().getSelectedIndex();
136             // if stepping through the list from above, skip to the succeeding item
                if ( (currentSelectedIndex == previousSelectedIndex + 1) )
138             {
                   getDocumentBox().setSelectedIndex( currentSelectedIndex + 1 );
140             }
                // if stepping through the list from below, skip to the preceeding item
142             else if ( currentSelectedIndex == previousSelectedIndex - 1 )
                {
144                getDocumentBox().setSelectedIndex( currentSelectedIndex - 1 );
                }
146             // if moving from non-adjacent item, negate selection
                else
148             {
                   getDocumentBox().setSelectedIndex( previousSelectedIndex );
150             }
             }
152          else
             {
154             previousSelectedIndex = getDocumentBox().getSelectedIndex();
             }
156       }
       }
158 };
    /** Custom renderer for the visible documents box. The box is filled with items of classes
160 * {@link #CaptionItem CaptionItem}, {@link #AuthorItem AuthorItem}, {@link #DocumentItem DocumentItem
        } which require
    * custom rendering to display with different levels of indentation and different icons.
162 */
    public class VisibleDocumentsBoxCellRenderer implements ListCellRenderer
164 {
       /** Render method for the visible documents list. Renders caption, author names and document names
             with proper indentation and icons
166    */
       public Component getListCellRendererComponent( JList list, Object value, int index, boolean
           isSelected, boolean cellHasFocus )
168    {
          JLabel renderLabel = new JLabel();
170       renderLabel.setIconTextGap( ClientConstants.getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() );
          renderLabel.setBorder( BorderFactory.createMatteBorder( 3, 3, 3, 3, renderLabel.getBackground()
                ));
172       renderLabel.setMaximumSize( ClientConstants.getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() );
          renderLabel.setPreferredSize( renderLabel.getMaximumSize() );
174       if ( value instanceof AuthorItem )
          {
176          renderLabel.setText( ( (AuthorItem) value ).getAuthorName() + ":");
             renderLabel.setIcon( new ImageIcon( getClass().getResource( "/toolbarButtonGraphics/
                 development/Host16.gif" )));
178       }
          else if ( value instanceof DocumentItem )
180       {
             DocumentItem docItem = (DocumentItem) value;
182          String docName = docItem.getDocumentName();
             if ( isSelected )
184          {
```

```
            renderLabel.setIcon( new javax.swing.ImageIcon( getClass().getResource( "/
                toolbarButtonGraphics/navigation/Forward16.gif") ));
            renderLabel.setForeground( ClientConstants.getVISIBLE_DOCUMENTS_SELECTION_COLOUR() );
         }
         renderLabel.setText( ClientConstants.getVISIBLE_DOCUMENTS_TAB_SIZE() +
                       docName );


      }
      // caption item last since it is the least likely (being unique)
      else if ( value instanceof CaptionItem )
      {
         renderLabel.setText( ( (CaptionItem) value ).getCaption() );
      // renderLabel.setIcon( new javax.swing.ImageIcon( getClass().getResource( "/
          toolbarButtonGraphics/navigation/Home16.gif" )));
         renderLabel.setHorizontalAlignment( SwingConstants.CENTER );
      // renderLabel.setBorder( BorderFactory.createMatteBorder( 6, 6, 6, 6, renderLabel.getBackground
          () ));
      }
      return renderLabel;
   }
};

/** Event handler for the message area at the bottom of the client application. */
public class MessageAreaMouseListener extends MouseAdapter
{
   public void mouseClicked( MouseEvent me )
   {
      {
         if ( me.getClickCount() == 2 )
         {
            showMessageWindow();
         }
      }
   }
}

/** Event handler for the client application main window */
public class ClientWindowListener extends WindowAdapter
{
   public void windowActivated( WindowEvent we )
   {
      bringChildWindowsOnTop();
   }
   public void windowClosing( WindowEvent we )
   {
       //System.out.println(" windowClosing ");
      try
      {
         terminateAndClose();
         dispose();
      }
      catch ( Throwable t )
      {
      }
   }
}
/** Event handler for the topic tree area of the client application main windows */
public class TopicTreeSelectionListener implements TreeSelectionListener
{
   public void valueChanged( TreeSelectionEvent tse )
   {
      handleTopicTreeClick( tse.getPath() );
   }
}
```

```
/** This is a small multi-purpose action listener which was implemented as an experiment to test
     whether running
* small tasks in the event-handling thread was feasible. It worked but interfered with the GUI.
*/
public class Task implements ActionListener
{
   private int currentValue;

   public Task(int startValue)
   {
      currentValue = startValue;
   }

   public void actionPerformed(ActionEvent actionEvent)
   {
      updateMessageArea(currentValue);
      // System.out.println("ACTION!");
   }

   public void updateValue(int newValue)
   {
      currentValue = newValue;
   }

   public void advance()
   {
      currentValue++;
   }
}

/** This class is used to represent clients that participate in JSDT sessions. It therefore
     implements the JSDT Client interface.
*/
public class DefaultClient implements Client
{
   String name;

   public DefaultClient (String nameArg )
   {
      name = nameArg;
   }
   public Object authenticate ( AuthenticationInfo ai )
   {
      return null;
   }
   public String getName ()
   {
      return name;
   }
}

public class TopicTreeMouseAdapter extends MouseAdapter
{
   public void mouseClicked( MouseEvent me )
   {
   // handleTopicTreeClick( me );
   }
}

/** The renderer class for the discussion tree in the client application main window. Leaf nodes in
     the tree are
* rendered with the string representation of the user object embedded in the node
*/
public class DiscussionTreeCellRenderer extends DefaultTreeCellRenderer
{

```

```
      public Component getTreeCellRendererComponent( JTree jTree, Object value, boolean selected,
           boolean expanded, boolean leaf, int row, boolean hasFocus )
312   {
         Component returnValue = super.getTreeCellRendererComponent( jTree, value, selected, expanded,
              leaf, row, hasFocus );
314      if ( leaf )
         {
316         if ( value instanceof Message)
            setText( value.toString() );
318      }
         return this;
320   }
   }
322
   /** The event handler associated with the compose new message button in the client application
        message area */
324 public class ComposeNewMessageButtonHandler implements ActionListener
   {
326   public void actionPerformed( ActionEvent ae )
      {
328      composeNewMessage();
      }
330 }

332 /** The event handler associated with the delete message button in the client application message
        area */
   public class DeleteButtonHandler implements ActionListener
334 {
      public void actionPerformed ( ActionEvent ae )
336   {
         deleteMessage();
338   }
   }
340
   /** The event handler associated with the message properties button in the client application message
         area */
342 public class MessagePropertiesButtonHandler implements ActionListener
   {
344   public void actionPerformed( ActionEvent ae )
      {
346      showMessageProperties();
      }
348 }
   private JMenuItem ivjAbout = null;
350 private AboutDialog ivjAboutDialog1 = null;
   private JMenuItem ivjAddDocuments = null;
352 private AddFiles ivjAddFiles1 = null;
   private JMenuBar ivjClientApplicationJMenuBar = null;
354 IvjEventHandler ivjEventHandler = new IvjEventHandler();
   private JMenuItem ivjExit = null;
356 private JButton ivjExitButton = null;
   private JMenu ivjHelpMenu = null;
358 private JMenuItem ivjHelpTopics = null;
   private JPanel ivjJAppletContentPane = null;
360 private JPanel ivjJPanel3 = null;
   private JSeparator ivjJSeparator10 = null;
362 private JSeparator ivjJSeparator11 = null;
   private JSeparator ivjJSeparator13 = null;
364 private JSeparator ivjJSeparator14 = null;
   private JSeparator ivjJSeparator2 = null;
366 private JSeparator ivjJSeparator5 = null;
   private JSeparator ivjJSeparator6 = null;
368 private JSeparator ivjJSeparator7 = null;
   private JSeparator ivjJSeparator9 = null;
370 private JToolBar ivjJToolBar1 = null;
   private JButton ivjJToolBarButton1 = null;
```

```
372     private JButton ivjJToolBarButton2 = null;
        private JButton ivjJToolBarButton3 = null;
374     private JPanel ivjMessagePanel = null;
        private BoxLayout ivjMessagePanelBoxLayout = null;
376     private ClientPanorama ivjPanoramaPanel = null;
        private JMenuItem ivjRemoveDocuments = null;
378     private RemoveFiles ivjRemoveFiles1 = null;
        private HelpDialog ivjHelpDialog1 = null;
380     private ClientManager clientManager;
        private JProgressBar ivjMessageArea = null;
382     private JButton ivjJToolBarButton5 = null;
        private JMenu ivjSetup = null;
384     private JButton ivjPreferencesButton = null;
        private JMenuItem ivjPreferencesItem = null;
386     private PreferencesDialog ivjPreferencesDialog1 = null;
        private JTree ivjDiscussionTree = null;
388     private JPanel ivjJPanel11 = null;
        private JPanel ivjJPanel21 = null;
390     private JTextPane ivjMessageDisplayArea = null;
        private JSplitPane ivjInnerSplitPane = null;
392     private JSplitPane ivjOuterSplitPane = null;
        private JToolBar ivjJToolBar2 = null;
394     private JMenu ivjFileMenu = null;
        private GroupDirectory ivjGroupDirectory1 = null;
396     private JMenuItem ivjGroups = null;
        private JButton ivjComposeNewMessageButton = null;
398     private JButton ivjMessagePropertiesButton = null;
        private JButton ivjDeleteButton = null;
400     private JScrollPane ivjJScrollPane1 = null;
        private boolean terminated = false;
402     private MessageWindow messageWindow = null;
        private ZComboBox ivjDocumentBox = null;
404
    class IvjEventHandler implements java.awt.event.ActionListener {
406         public void actionPerformed(java.awt.event.ActionEvent e) {
               if (e.getSource() == ClientApplication.this.getAbout())
408               connEtoM1(e);
               if (e.getSource() == ClientApplication.this.getAddDocuments())
410               connEtoM4(e);
               if (e.getSource() == ClientApplication.this.getRemoveDocuments())
412               connEtoM5(e);
               if (e.getSource() == ClientApplication.this.getExit())
414               connEtoM7(e);
               if (e.getSource() == ClientApplication.this.getJToolBarButton3())
416               connEtoM8(e);
               if (e.getSource() == ClientApplication.this.getJToolBarButton1())
418               connEtoM9(e);
               if (e.getSource() == ClientApplication.this.getPreferencesItem())
420               PreferencesMenuItemToDialog(e);
               if (e.getSource() == ClientApplication.this.getPreferencesButton())
422               PreferencesButtonToDialog(e);
               if (e.getSource() == ClientApplication.this.getGroups())
424               connEtoM2(e);
               if (e.getSource() == ClientApplication.this.getJToolBarButton5())
426               connEtoM3(e);
               if (e.getSource() == ClientApplication.this.getExitButton())
428               connEtoM6(e);
               if (e.getSource() == ClientApplication.this.getExit())
430               connEtoM10(e);
            };
432     };
        private java.util.Hashtable visibleDocumentsList = new Hashtable();
434     private javax.swing.Timer messageAreaTimer;
        private boolean itemListenerActive = true;
436 /**
     * ClientApplication constructor comment.
```

```
 */
public ClientApplication() {
   super();
   initialize();
}
/**
 * ClientApplication constructor comment.
 * @param title java.lang.String
 */
public ClientApplication(String title) {
   super(title);
}
/**
 * Add a new item to the list of visible documents found in the toolbar panel of the client application.
 * Creation date: (30-08-00 15:41:37)
 * @param senderName java.lang.String
 * @param authorName java.lang.String
 * @param documentName java.lang.String
 */
public void addDocumentToVisibleList(String senderName, String authorName, String documentNameAndPath,
     String documentName)
{
   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
   if ( getVisibleDocumentsList().containsKey( senderName ))
   {
      ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
      docList.add( new DocumentItem( fqPath, documentName ));
   }
   else
   {
      ArrayList docList = new ArrayList();
      docList.add( new AuthorItem( authorName ));
      docList.add( new DocumentItem( fqPath, documentName ));
      getVisibleDocumentsList().put( senderName, docList );
   }
}
/**
 * Advance the progress bar at the bottom of the client application by one step.
 * Creation date: (19-06-00 11:04:10)
 */
public void advanceTask()
{
   if ( ivjMessageArea.getValue() >= ivjMessageArea.getMaximum() )
   {
      initMessageAreaTimer();
      updateMessageArea( ivjMessageArea.getValue() );
   }
   else
   {
      updateMessageArea( ivjMessageArea.getValue()+1 );
   }
}
/**
 * Experimental code to ensure that child windows such as dialog boxes are always on top. This didn't to
       seem to work
 according to specifications, but I left the code here for future refinement.
 * Creation date: (16-08-00 15:32:07)
 */
public void bringChildWindowsOnTop()
{
   for ( int i = 0; i < getOwnedWindows().length; i++ )
   {
      getOwnedWindows()[i].toFront();
   }
}
/**
```

```
 * Center and size the window at predefined dimensions and position relative to the screen resolution
 * Creation date: (05-08-00 19:20:24)
 */
public void center()
{

   Dimension size;
   Dimension preferredSize = new Dimension(
                 Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                      getFRAME_HORIZONTAL_SIZE() )),
                 Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                      getFRAME_VERTICAL_SIZE() )));
   Dimension defaultSize = new Dimension(
                 Integer.parseInt( ClientConstants.getDEFAULT_FRAME_HORIZONTAL_SIZE() ),
                 Integer.parseInt( ClientConstants.getDEFAULT_FRAME_VERTICAL_SIZE() ));
   Point position;
   Point preferredPosition = new Point(
                 Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                      getFRAME_X_COORDINATE() )),
                 Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                      getFRAME_Y_COORDINATE() )));
   Point defaultPosition = new Point(
                 Integer.parseInt( ClientConstants.getDEFAULT_FRAME_X_COORDINATE() ),
                 Integer.parseInt( ClientConstants.getDEFAULT_FRAME_Y_COORDINATE() ));
   // if no value has overridden the default setting then query the OS for the dimensions of the display
   if ( preferredSize.equals( defaultSize ))
   {
      size = new Dimension(  (int) (getToolkit().getScreenSize().getWidth() * ClientConstants.
           getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
                       (int) (getToolkit().getScreenSize().getHeight() * ClientConstants.
                            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) );
   }
   else
   {
      size = preferredSize;
   }
   if ( preferredPosition.equals( defaultPosition ))
   {
      position = new Point(  (int)( getToolkit().getScreenSize().getWidth() * ( (1.0 - ClientConstants.
           getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2) ),
                       (int)( getToolkit().getScreenSize().getHeight() * ( (1.0 - ClientConstants.
                            getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2) ));
   }
   else
   {
      position = preferredPosition;
   }
   // System.out.println( size.toString() );
   setSize( size );
   setLocation( position );
   setState( java.awt.Frame.NORMAL );
   repaint();
}
/**
 * Check preconditions for opening a dialog box and, if fulfilled, open it, await completion and
     validation and then
 * insert the new message, if any, into the topic tree where indicated by the user.
 * Creation date: (10-07-00 22:00:41)
 */
public void composeNewMessage()
{
   if ( getClientManager().verifyGroupConnectionManagement() == false )
   {
      return;
   }
   if ( getDiscussionTree().getSelectionCount() == 0 )
```

```
    {
560    ErrorBox noSelectionErrorBox = new ErrorBox();
       noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
562    noSelectionErrorBox.show();
       return;
564 }
    else
566 {
       ComposeNewMessage composeNewMessage = new ComposeNewMessage();
568    composeNewMessage.setClientApplication( this );
       composeNewMessage.setAuthor( ClientInfo.getPreferences().getProperty( ClientConstants.
            getAUTHOR_NAME() ));
570    composeNewMessage.show();
       if ( composeNewMessage.getMessage() != null )
572    {
          // construct new message based on the user input in the dialog box and insert it into the topic
               tree.
574       Message message = composeNewMessage.getMessage();
          message.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() ));
576       message.setDocument( getClientManager().getClientInfo().getCurrentDocumentName() );

578       TreePath selectionPath = getDiscussionTree().getSelectionPath();
          DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectionPath.
               getLastPathComponent();
580       DefaultMutableTreeNode newTreeNode = new DefaultMutableTreeNode( message.getMessageID(), false )
               ;
          DefaultMutableTreeNode parentNode = null;
582       DefaultMutableTreeNode newNode = null;
          int insertionIndex = 0;
584       if ( selectedNode.isLeaf() && ( selectedNode.getParent() != null ))
          {
586          // create leaf node associated with new message
             parentNode = ( (DefaultMutableTreeNode) selectedNode.getParent() );
588          newNode = newTreeNode;
             // insert node immediately below the selected leaf node
590          insertionIndex = parentNode.getIndex( selectedNode )+ 1;
          }
592       else
          {
594          newNode = new DefaultMutableTreeNode( message.getTitle(), true );
             newNode.add( newTreeNode );
596          parentNode = selectedNode;
             // insert node below bottommost child of the selected node's parent
598          insertionIndex = parentNode.getChildCount();
          }
600       ( (DefaultTreeModel) getDiscussionTree().getModel() ).insertNodeInto( newNode, parentNode,
               parentNode.getChildCount() );
          //getDiscussionTree().scrollPathToVisible( new TreePath( newNode.getPath() ));
602       //getDiscussionTree().treeDidChange();
          getDiscussionTree().repaint();
604
          clientManager.sendMessage( message );
606       clientManager.sendTopicTreeUpdate( (DefaultTreeModel) getDiscussionTree().getModel() );
       }
608
    }
610 }
    /**
612  * connEtoM1: (About.action.actionPerformed(java.awt.event.ActionEvent) --> AboutDialog1.show()V)
     * @param arg1 java.awt.event.ActionEvent
614  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
616 private void connEtoM1(java.awt.event.ActionEvent arg1) {
       try {
618       // user code begin {1}
          // user code end
```

```
620         getAboutDialog1().show();
            // user code begin {2}
622         // user code end
        } catch (java.lang.Throwable ivjExc) {
624         // user code begin {3}
            // user code end
626         handleException(ivjExc);
        }
628 }
    /**
630  * connEtoM10: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
         terminateAndClose()V)
     * @param arg1 java.awt.event.ActionEvent
632  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
634 private void connEtoM10(java.awt.event.ActionEvent arg1) {
        try {
636         // user code begin {1}
            // user code end
638         this.terminateAndClose();
            // user code begin {2}
640         // user code end
        } catch (java.lang.Throwable ivjExc) {
642         // user code begin {3}
            // user code end
644         handleException(ivjExc);
        }
646 }
    /**
648  * connEtoM2: (Groups.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory1.show()V)
     * @param arg1 java.awt.event.ActionEvent
650  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
652 private void connEtoM2(java.awt.event.ActionEvent arg1) {
        try {
654         // user code begin {1}
            getGroupDirectory1().updateTable();
656         // user code end
            getGroupDirectory1().show();
658         // user code begin {2}
            // user code end
660     } catch (java.lang.Throwable ivjExc) {
            // user code begin {3}
662         // user code end
            handleException(ivjExc);
664     }
    }
666 /**
     * Insert the method's description here.
668  * Creation date: (27-06-00 16:58:47)
     * @param actionEvent java.awt.event.ActionEvent
670  */
    private void connEtoM20(ActionEvent actionEvent)
672 {
        try {
674         // user code begin {1}
            // user code end
676         getPreferencesDialog1().show();
            // user code begin {2}
678         // user code end
        } catch (java.lang.Throwable ivjExc) {
680         // user code begin {3}
            // user code end
682         handleException(ivjExc);
        }
684
```

```
    }
686 /**
     * connEtoM3: (JToolBarButton5.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory1.
         show()V)
688  * @param arg1 java.awt.event.ActionEvent
     */
690 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM3(java.awt.event.ActionEvent arg1) {
692    try {
          // user code begin {1}
694       // user code end
          getGroupDirectory1().show();
696       // user code begin {2}
          // user code end
698    } catch (java.lang.Throwable ivjExc) {
          // user code begin {3}
700       // user code end
          handleException(ivjExc);
702    }
    }
704 /**
     * connEtoM4: (AddDocuments.action.actionPerformed(java.awt.event.ActionEvent) --> AddFiles1.show()V)
706  * @param arg1 java.awt.event.ActionEvent
     */
708 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM4(java.awt.event.ActionEvent arg1) {
710    try {
          // user code begin {1}
712       // user code end
          getAddFiles1().show();
714       // user code begin {2}
          // user code end
716    } catch (java.lang.Throwable ivjExc) {
          // user code begin {3}
718       // user code end
          handleException(ivjExc);
720    }
    }
722 /**
     * connEtoM5: (RemoveDocuments.action.actionPerformed(java.awt.event.ActionEvent) --> RemoveFiles1.show
         ()V)
724  * @param arg1 java.awt.event.ActionEvent
     */
726 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM5(java.awt.event.ActionEvent arg1) {
728    try {
          // user code begin {1}
730
          // user code end
732       getRemoveFiles1().show();
          // user code begin {2}
734       // user code end
       } catch (java.lang.Throwable ivjExc) {
736       // user code begin {3}
          // user code end
738       handleException(ivjExc);
       }
740 }
    /**
742  * connEtoM6: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
         terminateAndClose()V)
     * @param arg1 java.awt.event.ActionEvent
744  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
746 private void connEtoM6(java.awt.event.ActionEvent arg1) {
       try {
```

```
748         // user code begin {1}
            // user code end
750         this.terminateAndClose();
            // user code begin {2}
752         // user code end
         } catch (java.lang.Throwable ivjExc) {
754         // user code begin {3}
            // user code end
756         handleException(ivjExc);
         }
758   }
      /**
760    * connEtoM7: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.dispose()V)
       * @param arg1 java.awt.event.ActionEvent
762    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
764   private void connEtoM7(java.awt.event.ActionEvent arg1) {
         try {
766         // user code begin {1}
            prepareTermination();
768         // user code end
            this.dispose();
770         // user code begin {2}
            // user code end
772      } catch (java.lang.Throwable ivjExc) {
            // user code begin {3}
774         // user code end
            handleException(ivjExc);
776      }
      }
778   /**
       * connEtoM8: (JToolBarButton3.action.actionPerformed(java.awt.event.ActionEvent) --> AddFiles1.show()V)
780    * @param arg1 java.awt.event.ActionEvent
       */
782   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private void connEtoM8(java.awt.event.ActionEvent arg1) {
784      try {
            // user code begin {1}
786         // user code end
            getAddFiles1().show();
788         // user code begin {2}
            // user code end
790      } catch (java.lang.Throwable ivjExc) {
            // user code begin {3}
792         // user code end
            handleException(ivjExc);
794      }
      }
796   /**
       * connEtoM9: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) --> RemoveFiles1.show
           ()V)
798    * @param arg1 java.awt.event.ActionEvent
       */
800   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private void connEtoM9(java.awt.event.ActionEvent arg1) {
802      try {
            // user code begin {1}
804         // user code end
            getRemoveFiles1().show();
806         // user code begin {2}
            // user code end
808      } catch (java.lang.Throwable ivjExc) {
            // user code begin {3}
810         // user code end
            handleException(ivjExc);
812      }
```

```
    }
814 /**
     * Deletes a message from the topic tree if one is selected and has not been replied to.
816  * Creation date: (12-07-00 10:23:34)
     */
818 public void deleteMessage()
    {
820   if ( getClientManager().verifyGroupConnectionManagement() == false )
      {
822     return;
      }
824   if ( getDiscussionTree().getSelectionCount() == 0 )
      {
826     ErrorBox noSelectionErrorBox = new ErrorBox();
        noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
828     noSelectionErrorBox.show();
      }
830   else
      {
832     TreePath selectedPath = getDiscussionTree().getSelectionPath();
        DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectedPath.getLastPathComponent()
            ;
834     // if the selected node is not a leaf, i.e. if it is an inner node, then exit immediately, without
            displaying an error message
        if ( selectedNode.isLeaf() == false)
836     {
          return;
838     }
        DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode) selectedNode.getParent();
840     DefaultMutableTreeNode successorNode = (DefaultMutableTreeNode) parentNode.getChildAfter(
            selectedNode );
        // if the selected node has not been responded to AND this client composed the message
842     if   ( ( ( successorNode == null) || ( successorNode.isLeaf() == false ))&&
            (getClientManager().getClientInfo().getCurrentMessage().getCreator().equalsIgnoreCase( (String
                ) ClientInfo.getPreferences().get( ClientConstants.getNAME() )))
844       )
        {
846       ConfirmationBox confirmationBox = new ConfirmationBox();
          confirmationBox.setText( ClientConstants.getCONFIRM_MESSAGE_DELETION( (String) getClientManager
              ().getClientInfo().getCurrentMessage().getTitle() ));
848       confirmationBox.show();
          if ( confirmationBox.isConfirmed() )
850       {
            // user has confirmed deletion, so proceed accordingly
852         clientManager.sendMessageDeletion( (String) selectedNode.getUserObject() );
            DefaultMutableTreeNode predecessorNode = (DefaultMutableTreeNode) parentNode.getChildBefore(
                selectedNode );
854         // if selectedNode is the first and only child of an inner node, then remove that also
            if ( predecessorNode == null )
856         {
              parentNode.removeFromParent();
858         }
            selectedNode.removeFromParent();
860         ( (DefaultTreeModel) getDiscussionTree().getModel() ).reload();
            clientManager.sendTopicTreeUpdate( (DefaultTreeModel) getDiscussionTree().getModel() );
862       }
        }
864     else
        {
866       ErrorBox invalidSelection = new ErrorBox();
          invalidSelection.setText( ClientConstants.getINVALID_DELETION_ATTEMPTED_MESSAGE() );
868       invalidSelection.show();
        }
870
      }
872 }
```

```
/**
 * Insert the method's description here.
 * Creation date: (28-07-00 00:27:12)
 */
protected void finalize() throws Throwable
{
   // System.out.println( " finalized ") ;
   // ensure proper termination
/* if ( isTerminated() == false )
   {
      terminateAndClose();
   }
*/
}
/**
 * Return the About property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getAbout() {
   if (ivjAbout == null) {
      try {
         ivjAbout = new javax.swing.JMenuItem();
         ivjAbout.setName("About");
         ivjAbout.setText("About");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAbout;
}
/**
 * Return the AboutDialog1 property value.
 * @return clientapp.AboutDialog
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private AboutDialog getAboutDialog1() {
   if (ivjAboutDialog1 == null) {
      try {
         ivjAboutDialog1 = new peerviewmisc.AboutDialog();
         ivjAboutDialog1.setName("AboutDialog1");
         ivjAboutDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAboutDialog1;
}
/**
 * Return the AddDocuments property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JMenuItem getAddDocuments() {
   if (ivjAddDocuments == null) {
      try {
         ivjAddDocuments = new javax.swing.JMenuItem();
         ivjAddDocuments.setName("AddDocuments");
```

```
            ivjAddDocuments.setText("Add␣Documents...");
940         // user code begin {1}
            // user code end
942      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
944         // user code end
            handleException(ivjExc);
946      }
         }
948      return ivjAddDocuments;
      }
950   /**
       * Return the AddFiles1 property value.
952    * @return clientapp.AddFiles
       */
954   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private AddFiles getAddFiles1() {
956      if (ivjAddFiles1 == null) {
            try {
958            ivjAddFiles1 = new clientapp.AddFiles();
               ivjAddFiles1.setName("AddFiles1");
960            ivjAddFiles1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
               // user code begin {1}
962            ivjAddFiles1.setClientApplication(this);
               // user code end
964         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
966            // user code end
               handleException(ivjExc);
968         }
         }
970      return ivjAddFiles1;
      }
972   /**
       * Return the ClientApplicationJMenuBar property value.
974    * @return javax.swing.JMenuBar
       */
976   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JMenuBar getClientApplicationJMenuBar() {
978      if (ivjClientApplicationJMenuBar == null) {
            try {
980            ivjClientApplicationJMenuBar = new javax.swing.JMenuBar();
               ivjClientApplicationJMenuBar.setName("ClientApplicationJMenuBar");
982            ivjClientApplicationJMenuBar.add(getFileMenu());
               ivjClientApplicationJMenuBar.add(getSetup());
984            ivjClientApplicationJMenuBar.add(getHelpMenu());
               // user code begin {1}
986            // user code end
            } catch (java.lang.Throwable ivjExc) {
988            // user code begin {2}
               // user code end
990            handleException(ivjExc);
            }
992      }
         return ivjClientApplicationJMenuBar;
994   }
      /**
996    * Return the client manager object associated with this client and create it if it doesn't already
           exist.
       * Creation date: (16-06-00 00:04:31)
998    * @return clientapp.ClientManager
       */
1000  public ClientManager getClientManager()
      {
1002     if ( clientManager == null )
         {
```

```
1004        clientManager = new ClientManager(this);
      }
1006    return clientManager;
    }
1008  /**
       * Return the DefaultToolBarButton property value.
1010   * @return javax.swing.JButton
       */
1012  /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getComposeNewMessageButton() {
1014    if (ivjComposeNewMessageButton == null) {
          try {
1016         ivjComposeNewMessageButton = new javax.swing.JButton();
             ivjComposeNewMessageButton.setName("ComposeNewMessageButton");
1018         ivjComposeNewMessageButton.setToolTipText("Compose␣new␣message");
             ivjComposeNewMessageButton.setMnemonic('C');
1020         ivjComposeNewMessageButton.setText("");
             ivjComposeNewMessageButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1022         ivjComposeNewMessageButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
             ivjComposeNewMessageButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                 toolbarButtonGraphics/general/ComposeMail16.gif")));
1024         ivjComposeNewMessageButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
1026         ivjComposeNewMessageButton.addActionListener( new ComposeNewMessageButtonHandler() );
             // user code end
1028      } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
1030         // user code end
             handleException(ivjExc);
1032      }
        }
1034    return ivjComposeNewMessageButton;
    }
1036  /**
       * Return the DeleteButton property value.
1038   * @return javax.swing.JButton
       */
1040  /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getDeleteButton() {
1042    if (ivjDeleteButton == null) {
          try {
1044         ivjDeleteButton = new javax.swing.JButton();
             ivjDeleteButton.setName("DeleteButton");
1046         ivjDeleteButton.setToolTipText("Delete␣message");
             ivjDeleteButton.setMnemonic('D');
1048         ivjDeleteButton.setText("");
             ivjDeleteButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1050         ivjDeleteButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
             ivjDeleteButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics
                 /general/Delete16.gif")));
1052         ivjDeleteButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
1054         ivjDeleteButton.addActionListener( new DeleteButtonHandler() );
             // user code end
1056      } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
1058         // user code end
             handleException(ivjExc);
1060      }
        }
1062    return ivjDeleteButton;
    }
1064  /**
       * Return the DiscussionTree property value. This method ought to be called getTopicTree() - TO DO
1066   * @return javax.swing.JTree
       */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTree getDiscussionTree() {
   if (ivjDiscussionTree == null) {
      try {
         ivjDiscussionTree = new javax.swing.JTree();
         ivjDiscussionTree.setName("DiscussionTree");
         ivjDiscussionTree.setToolTipText("Discussion␣tree");
         ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
         ivjDiscussionTree.setShowsRootHandles(true);
         ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 1));
         ivjDiscussionTree.setMinimumSize(new java.awt.Dimension(1, 1));
         // user code begin {1}
         DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode();
         ivjDiscussionTree.setModel( new DefaultTreeModel( rootNode ));
         ivjDiscussionTree.setRootVisible( true );
//       ivjDiscussionTree.setCellRenderer( new DiscussionTreeCellRenderer() );
         ivjDiscussionTree.addMouseListener( new TopicTreeMouseAdapter( ));
         ivjDiscussionTree.addTreeSelectionListener( new TopicTreeSelectionListener() );
         ivjDiscussionTree.getSelectionModel().setSelectionMode( TreeSelectionModel.SINGLE_TREE_SELECTION
               );
         ivjDiscussionTree.setShowsRootHandles( true );
         // make sure that the root is expanded
         ivjDiscussionTree.expandRow(0);
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjDiscussionTree;
}
/**
 * Return the DocumentBox property value.
 * @return javax.swing.JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JComboBox getDocumentBox() {
   if (ivjDocumentBox == null) {
      try {
         ivjDocumentBox = new ZComboBox();
         ivjDocumentBox.setName("DocumentBox");
         ivjDocumentBox.setToolTipText("Visible␣documents");
         // user code begin {1}
         ivjDocumentBox.setEditable( false );
         ivjDocumentBox.addItemListener( new VisibleDocumentsBoxItemListener() );
         ivjDocumentBox.setRenderer( new VisibleDocumentsBoxCellRenderer() );
         ivjDocumentBox.setMaximumRowCount( 15 );
         ivjDocumentBox.insertItemAt( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION()
               ), 0 );
         ivjDocumentBox.setSelectedIndex( 0 );
         ivjDocumentBox.setBorder( BorderFactory.createEtchedBorder() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjDocumentBox;
}
/**
 * Return the Exit property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
1132  private javax.swing.JMenuItem getExit() {
         if (ivjExit == null) {
1134        try {
               ivjExit = new javax.swing.JMenuItem();
1136           ivjExit.setName("Exit");
               ivjExit.setText("Exit");
1138           // user code begin {1}
               // user code end
1140        } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
1142           // user code end
               handleException(ivjExc);
1144        }
         }
1146     return ivjExit;
      }
1148  /**
       * Return the ExitButton property value.
1150   * @return javax.swing.JButton
       */
1152  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getExitButton() {
1154     if (ivjExitButton == null) {
         try {
1156        ivjExitButton = new javax.swing.JButton();
            ivjExitButton.setName("ExitButton");
1158        ivjExitButton.setToolTipText("Exit");
            ivjExitButton.setText("");
1160        ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1162        ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                general/Stop24.gif")));
            ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1164        // user code begin {1}
            // user code end
1166     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1168        // user code end
            handleException(ivjExc);
1170     }
         }
1172     return ivjExitButton;
      }
1174  /**
       * Return the DocumentMenu property value.
1176   * @return javax.swing.JMenu
       */
1178  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JMenu getFileMenu() {
1180     if (ivjFileMenu == null) {
         try {
1182        ivjFileMenu = new javax.swing.JMenu();
            ivjFileMenu.setName("FileMenu");
1184        ivjFileMenu.setMnemonic('F');
            ivjFileMenu.setText("File");
1186        ivjFileMenu.add(getAddDocuments());
            ivjFileMenu.add(getRemoveDocuments());
1188        ivjFileMenu.add(getJSeparator14());
            ivjFileMenu.add(getExit());
1190        // user code begin {1}
            // user code end
1192     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1194        // user code end
            handleException(ivjExc);
1196     }
```

```
      }
1198  return ivjFileMenu;
    }
1200 /**
     * Return the GroupDirectory1 property value.
1202 * @return clientapp.GroupDirectory
     */
1204 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    public GroupDirectory getGroupDirectory1() {
1206  if (ivjGroupDirectory1 == null) {
        try {
1208        ivjGroupDirectory1 = new clientapp.GroupDirectory();
            ivjGroupDirectory1.setName("GroupDirectory1");
1210        ivjGroupDirectory1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin {1}
1212        ivjGroupDirectory1.setClientApplication( this );
            ivjGroupDirectory1.initTable();
1214        // user code end
        } catch (java.lang.Throwable ivjExc) {
1216        // user code begin {2}
            // user code end
1218        handleException(ivjExc);
        }
1220  }
      return ivjGroupDirectory1;
1222 }
    /**
1224 * Return the Communication property value.
     * @return javax.swing.JMenuItem
1226 */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
1228 public javax.swing.JMenuItem getGroups() {
      if (ivjGroups == null) {
1230    try {
            ivjGroups = new javax.swing.JMenuItem();
1232        ivjGroups.setName("Groups");
            ivjGroups.setMnemonic('g');
1234        ivjGroups.setText("Groups");
            // user code begin {1}
1236        // user code end
        } catch (java.lang.Throwable ivjExc) {
1238        // user code begin {2}
            // user code end
1240        handleException(ivjExc);
        }
1242  }
      return ivjGroups;
1244 }
    /**
1246 * Return the HelpDialog1 property value.
     * @return peerviewmisc.HelpDialog
1248 */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
1250 private peerviewmisc.HelpDialog getHelpDialog1() {
      if (ivjHelpDialog1 == null) {
1252    try {
            ivjHelpDialog1 = new peerviewmisc.HelpDialog();
1254        ivjHelpDialog1.setName("HelpDialog1");
            ivjHelpDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1256        // user code begin {1}
            // user code end
1258    } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1260        // user code end
            handleException(ivjExc);
1262    }
```

```
      }
      return ivjHelpDialog1;
   }
   /**
    * Return the HelpMenu property value.
    * @return javax.swing.JMenu
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenu getHelpMenu() {
      if (ivjHelpMenu == null) {
         try {
            ivjHelpMenu = new javax.swing.JMenu();
            ivjHelpMenu.setName("HelpMenu");
            ivjHelpMenu.setText("Help");
            ivjHelpMenu.add(getHelpTopics());
            ivjHelpMenu.add(getAbout());
            // user code begin {1}
            // help system setup
            HelpSet hs;
            try
            {
               String helpPathAndName = ClientConstants.getCLIENT_DOCUMENTATION_HELPSET_NAME();
               java.net.URL hsURL = HelpSet.findHelpSet( null, helpPathAndName );
               hs = new HelpSet( null, hsURL );
               HelpBroker hb = hs.createHelpBroker();
               getHelpTopics().addActionListener( new HelpActionListener( hb ));
            }
            catch ( Exception E )
            {
               System.err.println( ClientConstants.getCOULD_NOT_INITIALIZE_HELP() );
            }
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjHelpMenu;
   }
   /**
    * Return the HelpTopics property value.
    * @return javax.swing.JMenuItem
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getHelpTopics() {
      if (ivjHelpTopics == null) {
         try {
            ivjHelpTopics = new javax.swing.JMenuItem();
            ivjHelpTopics.setName("HelpTopics");
            ivjHelpTopics.setText("Help␣Topics");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjHelpTopics;
   }
   /**
    * Return the JSplitPane11 property value.
    * @return javax.swing.JSplitPane
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
private javax.swing.JSplitPane getInnerSplitPane() {
   if (ivjInnerSplitPane == null) {
      try {
         ivjInnerSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.HORIZONTAL_SPLIT);
         ivjInnerSplitPane.setName("InnerSplitPane");
         ivjInnerSplitPane.setDividerSize(6);
         ivjInnerSplitPane.setMaximumSize(new java.awt.Dimension(100000, 100000));
         ivjInnerSplitPane.setDividerLocation(300);
         ivjInnerSplitPane.setPreferredSize(new java.awt.Dimension(1, 1));
         ivjInnerSplitPane.setOneTouchExpandable(true);
         ivjInnerSplitPane.setBounds(224, 808, 798, 503);
         ivjInnerSplitPane.setMinimumSize(new java.awt.Dimension(1, 1));
         getInnerSplitPane().add(getJPanel11(), "left");
         getInnerSplitPane().add(getJPanel21(), "right");
         // user code begin {1}
         ivjInnerSplitPane.resetToPreferredSizes();
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjInnerSplitPane;
}
/**
 * Return the JAppletContentPane property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJAppletContentPane() {
   if (ivjJAppletContentPane == null) {
      try {
         ivjJAppletContentPane = new javax.swing.JPanel();
         ivjJAppletContentPane.setName("JAppletContentPane");
         ivjJAppletContentPane.setLayout(new java.awt.BorderLayout());
         getJAppletContentPane().add(getMessagePanel(), "South");
         getJAppletContentPane().add(getJToolBar1(), "North");
         getJAppletContentPane().add(getOuterSplitPane(), "Center");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjJAppletContentPane;
}
/**
 * Return the JPanel11 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel11() {
   if (ivjJPanel11 == null) {
      try {
         ivjJPanel11 = new javax.swing.JPanel();
         ivjJPanel11.setName("JPanel11");
         ivjJPanel11.setPreferredSize(new java.awt.Dimension(150, 1));
         ivjJPanel11.setLayout(new java.awt.GridBagLayout());
         ivjJPanel11.setBackground(java.awt.SystemColor.window);
         ivjJPanel11.setMinimumSize(new java.awt.Dimension(1, 1));

         java.awt.GridBagConstraints constraintsDiscussionTree = new java.awt.GridBagConstraints();
         constraintsDiscussionTree.gridx = 0; constraintsDiscussionTree.gridy = 0;
```

```
            constraintsDiscussionTree.fill = java.awt.GridBagConstraints.BOTH;
1396        constraintsDiscussionTree.anchor = java.awt.GridBagConstraints.WEST;
            constraintsDiscussionTree.weightx = 1.0;
1398        constraintsDiscussionTree.weighty = 1.0;
            getJPanel11().add(getDiscussionTree(), constraintsDiscussionTree);
1400
            java.awt.GridBagConstraints constraintsJToolBar2 = new java.awt.GridBagConstraints();
1402        constraintsJToolBar2.gridx = 1; constraintsJToolBar2.gridy = 0;
            constraintsJToolBar2.fill = java.awt.GridBagConstraints.VERTICAL;
1404        constraintsJToolBar2.anchor = java.awt.GridBagConstraints.NORTHEAST;
            getJPanel11().add(getJToolBar2(), constraintsJToolBar2);
1406        // user code begin {1}
            // user code end
1408     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1410        // user code end
            handleException(ivjExc);
1412     }
         }
1414     return ivjJPanel11;
      }
1416  /**
       * Return the JPanel21 property value.
1418   * @return javax.swing.JPanel
       */
1420  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JPanel getJPanel21() {
1422     if (ivjJPanel21 == null) {
            try {
1424        ivjJPanel21 = new javax.swing.JPanel();
            ivjJPanel21.setName("JPanel21");
1426        ivjJPanel21.setPreferredSize(new java.awt.Dimension(450, 1));
            ivjJPanel21.setLayout(new java.awt.BorderLayout());
1428        ivjJPanel21.setMinimumSize(new java.awt.Dimension(1, 1));
            getJPanel21().add(getJScrollPane1(), "Center");
1430        // user code begin {1}
            // user code end
1432     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1434        // user code end
            handleException(ivjExc);
1436     }
         }
1438     return ivjJPanel21;
      }
1440  /**
       * Return the JPanel3 property value.
1442   * @return javax.swing.JPanel
       */
1444  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JPanel getJPanel3() {
1446     if (ivjJPanel3 == null) {
            try {
1448        ivjJPanel3 = new javax.swing.JPanel();
            ivjJPanel3.setName("JPanel3");
1450        ivjJPanel3.setLayout(new java.awt.GridBagLayout());

1452        java.awt.GridBagConstraints constraintsJSeparator11 = new java.awt.GridBagConstraints();
            constraintsJSeparator11.gridx = 2; constraintsJSeparator11.gridy = 1;
1454        constraintsJSeparator11.gridwidth = 0;
            constraintsJSeparator11.ipadx = -1;
1456        constraintsJSeparator11.ipady = 1;
            constraintsJSeparator11.insets = new java.awt.Insets(19, 0, 20, 0);
1458        getJPanel3().add(getJSeparator11(), constraintsJSeparator11);

1460        java.awt.GridBagConstraints constraintsJSeparator10 = new java.awt.GridBagConstraints();
```

```
        constraintsJSeparator10.gridx = 3; constraintsJSeparator10.gridy = 1;
1462    constraintsJSeparator10.gridwidth = 0;
        constraintsJSeparator10.ipadx = -1;
1464    constraintsJSeparator10.ipady = 1;
        constraintsJSeparator10.insets = new java.awt.Insets(19, 0, 20, 0);
1466    getJPanel3().add(getJSeparator10(), constraintsJSeparator10);

1468    java.awt.GridBagConstraints constraintsJToolBarButton3 = new java.awt.GridBagConstraints();
        constraintsJToolBarButton3.gridx = 3; constraintsJToolBarButton3.gridy = 1;
1470    constraintsJToolBarButton3.insets = new java.awt.Insets(5, 5, 5, 2);
        getJPanel3().add(getJToolBarButton3(), constraintsJToolBarButton3);
1472
        java.awt.GridBagConstraints constraintsJToolBarButton1 = new java.awt.GridBagConstraints();
1474    constraintsJToolBarButton1.gridx = 4; constraintsJToolBarButton1.gridy = 1;
        constraintsJToolBarButton1.insets = new java.awt.Insets(5, 3, 5, 5);
1476    getJPanel3().add(getJToolBarButton1(), constraintsJToolBarButton1);

1478    java.awt.GridBagConstraints constraintsJSeparator2 = new java.awt.GridBagConstraints();
        constraintsJSeparator2.gridx = 5; constraintsJSeparator2.gridy = 1;
1480    constraintsJSeparator2.gridwidth = 0;
        constraintsJSeparator2.ipadx = -1;
1482    constraintsJSeparator2.ipady = 1;
        constraintsJSeparator2.insets = new java.awt.Insets(19, 0, 20, 0);
1484    getJPanel3().add(getJSeparator2(), constraintsJSeparator2);

1486    java.awt.GridBagConstraints constraintsJSeparator5 = new java.awt.GridBagConstraints();
        constraintsJSeparator5.gridx = 6; constraintsJSeparator5.gridy = 1;
1488    constraintsJSeparator5.gridwidth = 0;
        constraintsJSeparator5.ipadx = -1;
1490    constraintsJSeparator5.ipady = 1;
        constraintsJSeparator5.insets = new java.awt.Insets(19, 0, 20, 0);
1492    getJPanel3().add(getJSeparator5(), constraintsJSeparator5);

1494    java.awt.GridBagConstraints constraintsJToolBarButton5 = new java.awt.GridBagConstraints();
        constraintsJToolBarButton5.gridx = 6; constraintsJToolBarButton5.gridy = 1;
1496    constraintsJToolBarButton5.insets = new java.awt.Insets(5, 5, 5, 2);
        getJPanel3().add(getJToolBarButton5(), constraintsJToolBarButton5);
1498
        java.awt.GridBagConstraints constraintsPreferencesButton = new java.awt.GridBagConstraints();
1500    constraintsPreferencesButton.gridx = 7; constraintsPreferencesButton.gridy = 1;
        constraintsPreferencesButton.insets = new java.awt.Insets(5, 3, 5, 5);
1502    getJPanel3().add(getPreferencesButton(), constraintsPreferencesButton);

1504    java.awt.GridBagConstraints constraintsJSeparator13 = new java.awt.GridBagConstraints();
        constraintsJSeparator13.gridx = 8; constraintsJSeparator13.gridy = 1;
1506    constraintsJSeparator13.gridwidth = 0;
        constraintsJSeparator13.ipadx = -1;
1508    constraintsJSeparator13.ipady = 1;
        constraintsJSeparator13.insets = new java.awt.Insets(19, 0, 20, 0);
1510    getJPanel3().add(getJSeparator13(), constraintsJSeparator13);

1512    java.awt.GridBagConstraints constraintsJSeparator7 = new java.awt.GridBagConstraints();
        constraintsJSeparator7.gridx = 9; constraintsJSeparator7.gridy = 1;
1514    constraintsJSeparator7.gridwidth = 0;
        constraintsJSeparator7.ipadx = -1;
1516    constraintsJSeparator7.ipady = 1;
        constraintsJSeparator7.insets = new java.awt.Insets(19, 0, 20, 0);
1518    getJPanel3().add(getJSeparator7(), constraintsJSeparator7);

1520    java.awt.GridBagConstraints constraintsJToolBarButton2 = new java.awt.GridBagConstraints();
        constraintsJToolBarButton2.gridx = 9; constraintsJToolBarButton2.gridy = 1;
1522    constraintsJToolBarButton2.insets = new java.awt.Insets(5, 5, 5, 2);
        getJPanel3().add(getJToolBarButton2(), constraintsJToolBarButton2);
1524
        java.awt.GridBagConstraints constraintsExitButton = new java.awt.GridBagConstraints();
1526    constraintsExitButton.gridx = 10; constraintsExitButton.gridy = 1;
```

```
            constraintsExitButton.insets = new java.awt.Insets(5, 3, 5, 5);
1528        getJPanel3().add(getExitButton(), constraintsExitButton);

1530        java.awt.GridBagConstraints constraintsJSeparator9 = new java.awt.GridBagConstraints();
            constraintsJSeparator9.gridx = 11; constraintsJSeparator9.gridy = 1;
1532        constraintsJSeparator9.gridwidth = 0;
            constraintsJSeparator9.ipadx = -1;
1534        constraintsJSeparator9.ipady = 1;
            constraintsJSeparator9.insets = new java.awt.Insets(19, 0, 20, 0);
1536        getJPanel3().add(getJSeparator9(), constraintsJSeparator9);

1538        java.awt.GridBagConstraints constraintsJSeparator6 = new java.awt.GridBagConstraints();
            constraintsJSeparator6.gridx = 12; constraintsJSeparator6.gridy = 1;
1540        constraintsJSeparator6.weightx = 1.0;
            constraintsJSeparator6.gridwidth = 0;
1542        constraintsJSeparator6.ipadx = -1;
            constraintsJSeparator6.ipady = 1;
1544        constraintsJSeparator6.insets = new java.awt.Insets(19, 0, 20, 0);
            getJPanel3().add(getJSeparator6(), constraintsJSeparator6);
1546
            java.awt.GridBagConstraints constraintsDocumentBox = new java.awt.GridBagConstraints();
1548        constraintsDocumentBox.gridx = 13; constraintsDocumentBox.gridy = 1;
            constraintsDocumentBox.anchor = java.awt.GridBagConstraints.EAST;
1550        constraintsDocumentBox.weightx = 0.0;
            constraintsDocumentBox.weighty = 0.0;
1552        constraintsDocumentBox.ipadx = 250;
            constraintsDocumentBox.ipady = 10;
1554        constraintsDocumentBox.insets = new java.awt.Insets(7, 5, 7, 7);
            getJPanel3().add(getDocumentBox(), constraintsDocumentBox);
1556        // user code begin {1}
            // user code end
1558     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1560        // user code end
            handleException(ivjExc);
1562     }
         }
1564     return ivjJPanel3;
      }
1566  /**
       * Return the JScrollPane1 property value.
1568   * @return javax.swing.JScrollPane
       */
1570  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JScrollPane getJScrollPane1() {
1572     if (ivjJScrollPane1 == null) {
            try {
1574        ivjJScrollPane1 = new javax.swing.JScrollPane();
            ivjJScrollPane1.setName("JScrollPane1");
1576        getJScrollPane1().setViewportView(getMessageDisplayArea());
            // user code begin {1}
1578        // user code end
         } catch (java.lang.Throwable ivjExc) {
1580        // user code begin {2}
            // user code end
1582        handleException(ivjExc);
         }
1584     }
         return ivjJScrollPane1;
1586  }
      /**
1588   * Return the JSeparator10 property value.
       * @return javax.swing.JSeparator
1590   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1592  private javax.swing.JSeparator getJSeparator10() {
```

```
      if (ivjJSeparator10 == null) {
1594     try {
            ivjJSeparator10 = new javax.swing.JSeparator();
1596        ivjJSeparator10.setName("JSeparator10");
            // user code begin {1}
1598        // user code end
         } catch (java.lang.Throwable ivjExc) {
1600        // user code begin {2}
            // user code end
1602        handleException(ivjExc);
         }
1604  }
      return ivjJSeparator10;
1606 }
     /**
1608  * Return the JSeparator11 property value.
      * @return javax.swing.JSeparator
1610  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1612 private javax.swing.JSeparator getJSeparator11() {
      if (ivjJSeparator11 == null) {
1614     try {
            ivjJSeparator11 = new javax.swing.JSeparator();
1616        ivjJSeparator11.setName("JSeparator11");
            // user code begin {1}
1618        // user code end
         } catch (java.lang.Throwable ivjExc) {
1620        // user code begin {2}
            // user code end
1622        handleException(ivjExc);
         }
1624  }
      return ivjJSeparator11;
1626 }
     /**
1628  * Return the JSeparator13 property value.
      * @return javax.swing.JSeparator
1630  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1632 private javax.swing.JSeparator getJSeparator13() {
      if (ivjJSeparator13 == null) {
1634     try {
            ivjJSeparator13 = new javax.swing.JSeparator();
1636        ivjJSeparator13.setName("JSeparator13");
            ivjJSeparator13.setFont(new java.awt.Font("dialog", 0, 24));
1638        // user code begin {1}
            // user code end
1640     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1642        // user code end
            handleException(ivjExc);
1644     }
      }
1646  return ivjJSeparator13;
     }
1648 /**
      * Return the JSeparator14 property value.
1650  * @return javax.swing.JSeparator
      */
1652 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JSeparator getJSeparator14() {
1654  if (ivjJSeparator14 == null) {
         try {
1656        ivjJSeparator14 = new javax.swing.JSeparator();
            ivjJSeparator14.setName("JSeparator14");
1658        // user code begin {1}
```

```
            // user code end
1660        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1662        // user code end
            handleException(ivjExc);
1664        }
         }
1666     return ivjJSeparator14;
      }
1668  /**
       * Return the JSeparator2 property value.
1670   * @return javax.swing.JSeparator
       */
1672  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JSeparator getJSeparator2() {
1674     if (ivjJSeparator2 == null) {
            try {
1676        ivjJSeparator2 = new javax.swing.JSeparator();
            ivjJSeparator2.setName("JSeparator2");
1678        // user code begin {1}
            // user code end
1680        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1682        // user code end
            handleException(ivjExc);
1684        }
         }
1686     return ivjJSeparator2;
      }
1688  /**
       * Return the JSeparator5 property value.
1690   * @return javax.swing.JSeparator
       */
1692  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JSeparator getJSeparator5() {
1694     if (ivjJSeparator5 == null) {
            try {
1696        ivjJSeparator5 = new javax.swing.JSeparator();
            ivjJSeparator5.setName("JSeparator5");
1698        // user code begin {1}
            // user code end
1700        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1702        // user code end
            handleException(ivjExc);
1704        }
         }
1706     return ivjJSeparator5;
      }
1708  /**
       * Return the JSeparator6 property value.
1710   * @return javax.swing.JSeparator
       */
1712  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JSeparator getJSeparator6() {
1714     if (ivjJSeparator6 == null) {
            try {
1716        ivjJSeparator6 = new javax.swing.JSeparator();
            ivjJSeparator6.setName("JSeparator6");
1718        // user code begin {1}
            // user code end
1720        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1722        // user code end
            handleException(ivjExc);
1724        }
```

```
      }
      return ivjJSeparator6;
   }
   /**
    * Return the JSeparator7 property value.
    * @return javax.swing.JSeparator
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JSeparator getJSeparator7() {
      if (ivjJSeparator7 == null) {
         try {
            ivjJSeparator7 = new javax.swing.JSeparator();
            ivjJSeparator7.setName("JSeparator7");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJSeparator7;
   }
   /**
    * Return the JSeparator9 property value.
    * @return javax.swing.JSeparator
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JSeparator getJSeparator9() {
      if (ivjJSeparator9 == null) {
         try {
            ivjJSeparator9 = new javax.swing.JSeparator();
            ivjJSeparator9.setName("JSeparator9");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJSeparator9;
   }
   /**
    * Return the JToolBar1 property value.
    * @return javax.swing.JToolBar
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JToolBar getJToolBar1() {
      if (ivjJToolBar1 == null) {
         try {
            ivjJToolBar1 = new javax.swing.JToolBar();
            ivjJToolBar1.setName("JToolBar1");
            ivjJToolBar1.setFloatable(false);
            getJToolBar1().add(getJPanel3(), getJPanel3().getName());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJToolBar1;
   }
   /**
```

```
     * Return the JToolBar2 property value.
1792  * @return javax.swing.JToolBar
     */
1794 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JToolBar getJToolBar2() {
1796    if (ivjJToolBar2 == null) {
           try {
1798          ivjJToolBar2 = new javax.swing.JToolBar();
              ivjJToolBar2.setName("JToolBar2");
1800          ivjJToolBar2.setFloatable(false);
              ivjJToolBar2.setBackground(java.awt.Color.white);
1802          ivjJToolBar2.setOrientation(javax.swing.SwingConstants.VERTICAL);
              ivjJToolBar2.add(getComposeNewMessageButton());
1804          getJToolBar2().add(getDeleteButton(), getDeleteButton().getName());
              getJToolBar2().add(getMessagePropertiesButton(), getMessagePropertiesButton().getName());
1806          // user code begin {1}
              // user code end
           } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1810          // user code end
              handleException(ivjExc);
1812       }
        }
1814    return ivjJToolBar2;
     }
1816 /**
     * Return the JToolBarButton1 property value.
1818  * @return javax.swing.JButton
     */
1820 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getJToolBarButton1() {
1822    if (ivjJToolBarButton1 == null) {
           try {
1824          ivjJToolBarButton1 = new javax.swing.JButton();
              ivjJToolBarButton1.setName("JToolBarButton1");
1826          ivjJToolBarButton1.setToolTipText("Remove␣Documents");
              ivjJToolBarButton1.setText("");
1828          ivjJToolBarButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
              ivjJToolBarButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1830          ivjJToolBarButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Export24.gif")));
              ivjJToolBarButton1.setMargin(new java.awt.Insets(0, 0, 0, 0));
1832          // user code begin {1}
              // user code end
           } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1836          // user code end
              handleException(ivjExc);
1838       }
        }
1840    return ivjJToolBarButton1;
     }
1842 /**
     * Return the JToolBarButton2 property value.
1844  * @return javax.swing.JButton
     */
1846 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getJToolBarButton2() {
1848    if (ivjJToolBarButton2 == null) {
           try {
1850          ivjJToolBarButton2 = new javax.swing.JButton();
              ivjJToolBarButton2.setName("JToolBarButton2");
1852          ivjJToolBarButton2.setToolTipText("Help");
              ivjJToolBarButton2.setText("");
1854          ivjJToolBarButton2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
              ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
```

```
1856          ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Help24.gif")));
              ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
1858          // user code begin {1}
              // user code end
1860       } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1862          // user code end
              handleException(ivjExc);
1864       }
        }
1866    return ivjJToolBarButton2;
     }
1868 /**
      * Return the JToolBarButton3 property value.
1870  * @return javax.swing.JButton
      */
1872 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getJToolBarButton3() {
1874    if (ivjJToolBarButton3 == null) {
        try {
1876          ivjJToolBarButton3 = new javax.swing.JButton();
              ivjJToolBarButton3.setName("JToolBarButton3");
1878          ivjJToolBarButton3.setToolTipText( "Add␣documents" );
              ivjJToolBarButton3.setText("");
1880          ivjJToolBarButton3.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
              ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1882          ivjJToolBarButton3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Import24.gif")));
              ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
1884          // user code begin {1}
              // user code end
1886       } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1888          // user code end
              handleException(ivjExc);
1890       }
        }
1892    return ivjJToolBarButton3;
     }
1894 /**
      * Return the JToolBarButton5 property value.
1896  * @return javax.swing.JButton
      */
1898 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getJToolBarButton5() {
1900    if (ivjJToolBarButton5 == null) {
        try {
1902          ivjJToolBarButton5 = new javax.swing.JButton();
              ivjJToolBarButton5.setName("JToolBarButton5");
1904          ivjJToolBarButton5.setText("");
              ivjJToolBarButton5.setToolTipText( "Group␣directory" );
1906          ivjJToolBarButton5.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
              ivjJToolBarButton5.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1908          ivjJToolBarButton5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Preferences24.gif")));
              ivjJToolBarButton5.setMargin(new java.awt.Insets(0, 0, 0, 0));
1910          // user code begin {1}
              // user code end
1912       } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1914          // user code end
              handleException(ivjExc);
1916       }
        }
1918    return ivjJToolBarButton5;
```

```
    }
1920 /**
     * Return the JProgressBar1 property value.
1922 * @return javax.swing.JProgressBar
     */
1924 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     public javax.swing.JProgressBar getMessageArea() {
1926    if (ivjMessageArea == null) {
           try {
1928          ivjMessageArea = new javax.swing.JProgressBar();
              ivjMessageArea.setName("MessageArea");
1930          ivjMessageArea.setString("Message␣area:␣double-click␣here␣to␣open␣in␣separate␣window...");
              ivjMessageArea.setStringPainted(true);
1932          // user code begin {1}
              ivjMessageArea.setPreferredSize( new Dimension( Integer.MAX_VALUE, (int) ivjMessageArea.
                  getPreferredSize().getHeight() ));
1934          ivjMessageArea.setMaximumSize( ivjMessageArea.getPreferredSize() );
              ivjMessageArea.addMouseListener( new MessageAreaMouseListener() );
1936          // user code end
           } catch (java.lang.Throwable ivjExc) {
1938          // user code begin {2}
              // user code end
1940          handleException(ivjExc);
           }
1942    }
        return ivjMessageArea;
1944 }
     /**
1946 * Return the ButtonPanelFlowLayout property value.
     * @return java.awt.FlowLayout
1948 */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1950 private java.awt.FlowLayout getMessageAreaFlowLayout() {
        java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
1952    try {
           /* Create part */
1954       ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
           ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
1956    } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
1958    };
        return ivjButtonPanelFlowLayout;
1960 }
     /**
1962 * Insert the method's description here.
     * Creation date: (31-08-00 02:18:56)
1964 * @return javax.swing.Timer
     */
1966 public javax.swing.Timer getMessageAreaTimer() {
        return messageAreaTimer;
1968 }
     /**
1970 * Return the MessageDisplayArea property value.
     * @return javax.swing.JTextPane
1972 */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1974 private javax.swing.JTextPane getMessageDisplayArea() {
        if (ivjMessageDisplayArea == null) {
1976       try {
              ivjMessageDisplayArea = new javax.swing.JTextPane();
1978          ivjMessageDisplayArea.setName("MessageDisplayArea");
              ivjMessageDisplayArea.setToolTipText("null");
1980          ivjMessageDisplayArea.setPreferredSize(new java.awt.Dimension(7, 1));
              ivjMessageDisplayArea.setBounds(0, 0, 490, 498);
1982          ivjMessageDisplayArea.setEnabled(false);
              ivjMessageDisplayArea.setMinimumSize(new java.awt.Dimension(6, 1));
```

```
1984          // user code begin {1}
              ivjMessageDisplayArea.setToolTipText(null);
1986          // user code end
           } catch (java.lang.Throwable ivjExc) {
1988          // user code begin {2}
              // user code end
1990          handleException(ivjExc);
           }
1992     }
         return ivjMessageDisplayArea;
1994  }
      /**
1996   * Return the MessagePanel property value.
       * @return javax.swing.JPanel
1998   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
2000  private javax.swing.JPanel getMessagePanel() {
         if (ivjMessagePanel == null) {
2002        try {
              ivjMessagePanel = new javax.swing.JPanel();
2004          ivjMessagePanel.setName("MessagePanel");
              ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
2006
              getMessagePanel().add(getMessageArea(), getMessageArea().getName());
2008          // user code begin {1}
              // user code end
2010        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
2012          // user code end
              handleException(ivjExc);
2014        }
         }
2016     return ivjMessagePanel;
      }
2018  /**
       * Return the MessagePanelBoxLayout property value.
2020   * @return javax.swing.BoxLayout
       */
2022  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.BoxLayout getMessagePanelBoxLayout() {
2024     javax.swing.BoxLayout ivjMessagePanelBoxLayout = null;
         try {
2026        /* Create part */
            ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swing.BoxLayout.
                Y_AXIS);
2028     } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
2030     };
         return ivjMessagePanelBoxLayout;
2032  }
      /**
2034   * Return the JToolBarButton4 property value.
       * @return javax.swing.JButton
2036   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
2038  private javax.swing.JButton getMessagePropertiesButton() {
         if (ivjMessagePropertiesButton == null) {
2040        try {
              ivjMessagePropertiesButton = new javax.swing.JButton();
2042          ivjMessagePropertiesButton.setName("MessagePropertiesButton");
              ivjMessagePropertiesButton.setToolTipText("View␣message␣properties");
2044          ivjMessagePropertiesButton.setMnemonic('V');
              ivjMessagePropertiesButton.setText("");
2046          ivjMessagePropertiesButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
              ivjMessagePropertiesButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
```

```
2048          ivjMessagePropertiesButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Information16.gif")));
              ivjMessagePropertiesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
2050          // user code begin {1}
              ivjMessagePropertiesButton.addActionListener( new MessagePropertiesButtonHandler() );
2052          // user code end
           } catch (java.lang.Throwable ivjExc) {
2054          // user code begin {2}
              // user code end
2056          handleException(ivjExc);
           }
2058     }
         return ivjMessagePropertiesButton;
2060  }
      /**
2062   * Insert the method's description here.
       * Creation date: (16-08-00 17:05:09)
2064   * @return clientapp.MessageWindow
       */
2066  public MessageWindow getMessageWindow()
      {
2068     if ( messageWindow == null )
         {
2070        messageWindow = new MessageWindow();
         }
2072     return messageWindow;
      }
2074  /**
       * Return the JSplitPane1 property value.
2076   * @return javax.swing.JSplitPane
       */
2078  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JSplitPane getOuterSplitPane() {
2080     if (ivjOuterSplitPane == null) {
            try {
2082           ivjOuterSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
               ivjOuterSplitPane.setName("OuterSplitPane");
2084           ivjOuterSplitPane.setDividerSize(6);
               ivjOuterSplitPane.setLastDividerLocation(600);
2086           ivjOuterSplitPane.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
               ivjOuterSplitPane.setDividerLocation(600);
2088           ivjOuterSplitPane.setPreferredSize(new java.awt.Dimension(589, 1000));
               ivjOuterSplitPane.setAlignmentX(java.awt.Component.RIGHT_ALIGNMENT);
2090           ivjOuterSplitPane.setOneTouchExpandable(true);
               ivjOuterSplitPane.setMinimumSize(new java.awt.Dimension(503, 1000));
2092           getOuterSplitPane().add(getPanoramaPanel(), "top");
               // user code begin {1}
2094           ivjOuterSplitPane.setDividerLocation( 0.8 );
               // user code end
2096        } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
2098           // user code end
               handleException(ivjExc);
2100        }
         }
2102     return ivjOuterSplitPane;
      }
2104  /**
       * Insert the method's description here.
2106   * Creation date: (04-08-00 14:39:32)
       * @return clientapp.ClientPanorama
2108   */
      public ClientPanorama getPanoramaPanel() {
2110     if (ivjPanoramaPanel == null) {
            try {
2112           ivjPanoramaPanel = new clientapp.ClientPanorama();
```

```
            ivjPanoramaPanel.setName("PanoramaPanel");
2114        ivjPanoramaPanel.setToolTipText("null");
            ivjPanoramaPanel.setPreferredSize(new java.awt.Dimension(300, 300));
2116        ivjPanoramaPanel.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
            ivjPanoramaPanel.setMinimumSize(new java.awt.Dimension(1, 1));
2118        // user code begin {1}
            // user code end
2120     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
2122        // user code end
            handleException(ivjExc);
2124     }
         }
2126     return ivjPanoramaPanel;
      }
2128  /**
       * Return the JToolBarButton4 property value.
2130   * @return javax.swing.JButton
       */
2132  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getPreferencesButton() {
2134     if (ivjPreferencesButton == null) {
         try {
2136        ivjPreferencesButton = new javax.swing.JButton();
            ivjPreferencesButton.setName("PreferencesButton");
2138        ivjPreferencesButton.setText("");
            ivjPreferencesButton.setToolTipText( "Preferences" );
2140        ivjPreferencesButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjPreferencesButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
2142        ivjPreferencesButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                toolbarButtonGraphics/development/Host24.gif")));
            ivjPreferencesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
2144        // user code begin {1}
            // user code end
2146     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
2148        // user code end
            handleException(ivjExc);
2150     }
         }
2152     return ivjPreferencesButton;
      }
2154  /**
       * Return the PreferencesDialog1 property value.
2156   * @return clientapp.PreferencesDialog
       */
2158  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private PreferencesDialog getPreferencesDialog1() {
2160     if (ivjPreferencesDialog1 == null) {
         try {
2162        ivjPreferencesDialog1 = new clientapp.PreferencesDialog();
            ivjPreferencesDialog1.setName("PreferencesDialog1");
2164        ivjPreferencesDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin {1}
2166        ivjPreferencesDialog1.setClientApplication( this );
            ivjPreferencesDialog1.init();
2168        // user code end
         } catch (java.lang.Throwable ivjExc) {
2170        // user code begin {2}
            // user code end
2172        handleException(ivjExc);
         }
2174     }
         return ivjPreferencesDialog1;
2176  }
      /**
```

```
 * Return the OptionsItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getPreferencesItem() {
   if (ivjPreferencesItem == null) {
      try {
         ivjPreferencesItem = new javax.swing.JMenuItem();
         ivjPreferencesItem.setName("PreferencesItem");
         ivjPreferencesItem.setText("Preferences");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjPreferencesItem;
}
/**
 * Return the RemoveDocuments property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JMenuItem getRemoveDocuments() {
   if (ivjRemoveDocuments == null) {
      try {
         ivjRemoveDocuments = new javax.swing.JMenuItem();
         ivjRemoveDocuments.setName("RemoveDocuments");
         ivjRemoveDocuments.setText("Remove␣Documents...");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjRemoveDocuments;
}
/**
 * Return the RemoveFiles1 property value.
 * @return clientapp.RemoveFiles
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private RemoveFiles getRemoveFiles1() {
   if (ivjRemoveFiles1 == null) {
      try {
         ivjRemoveFiles1 = new clientapp.RemoveFiles();
         ivjRemoveFiles1.setName("RemoveFiles1");
         // user code begin {1}
         ivjRemoveFiles1.setClientApplication( this );
         ivjRemoveFiles1.initialize();
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjRemoveFiles1;
}
/**
 * Return the Setup property value.
 * @return javax.swing.JMenu
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenu getSetup() {
   if (ivjSetup == null) {
      try {
         ivjSetup = new javax.swing.JMenu();
         ivjSetup.setName("Setup");
         ivjSetup.setMnemonic('S');
         ivjSetup.setText("Setup");
         ivjSetup.add(getGroups());
         ivjSetup.add(getPreferencesItem());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjSetup;
}
/**
 * Return the visibleDocumentsList property.
 * Creation date: (30-08-00 16:14:33)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getVisibleDocumentsList() {
   return visibleDocumentsList;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
    System.out.println("---------␣UNCAUGHT␣EXCEPTION␣---------");
    exception.printStackTrace(System.out);
}
/**
 * Insert the method's description here.
 * Creation date: (11-08-00 17:57:58)
 * @param mouseEvent java.awt.event.MouseEvent
 */
public void handleTopicTreeClick( TreePath selPath )
{
   if ( selPath != null )
   {
      DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selPath.getLastPathComponent();
      if ( treeNode.isLeaf() == true && treeNode.getUserObject() != null )
      {
         // retrieve ID for the message object associated with the selected node and request it from the
              server
         String messageID = (String) treeNode.getUserObject();
         getClientManager().requestMessageFromServer( messageID );
         // System.out.println( "Sendte bud efter " + messageID );
      }
      else
      {

         setMessage( null );
         getClientManager().getClientInfo().setCurrentMessage( null );
      }
   }
}
/**
```

```
     * Initializes connections
2310  * @exception java.lang.Exception The exception description.
     */
2312 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initConnections() throws java.lang.Exception {
2314    // user code begin {1}
        // user code end
2316    getAbout().addActionListener(ivjEventHandler);
        getAddDocuments().addActionListener(ivjEventHandler);
2318    getRemoveDocuments().addActionListener(ivjEventHandler);
        getExit().addActionListener(ivjEventHandler);
2320    getJToolBarButton3().addActionListener(ivjEventHandler);
        getJToolBarButton1().addActionListener(ivjEventHandler);
2322    getHelpTopics().addActionListener(ivjEventHandler);
        getJToolBarButton2().addActionListener(ivjEventHandler);
2324    getPreferencesItem().addActionListener(ivjEventHandler);
        getPreferencesButton().addActionListener(ivjEventHandler);
2326    getGroups().addActionListener(ivjEventHandler);
        getJToolBarButton5().addActionListener(ivjEventHandler);
2328    getExitButton().addActionListener(ivjEventHandler);
     }
2330 /**
     * Initialize the class.
2332 */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
2334 private void initialize()
     {
2336    try {
           // user code begin {1}
2338       initPanoramaPanel();
           // user code end
2340       setName("ClientApplication");
           setTitle("PeerView␣client");
2342       setSize(800, 600);
           setJMenuBar(getClientApplicationJMenuBar());
2344       setContentPane(getJAppletContentPane());
           initConnections();
2346    } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
2348    }
        // user code begin {2}
2350    setDefaultCloseOperation( WindowConstants.DO_NOTHING_ON_CLOSE );
        addWindowListener( new ClientWindowListener() );
2352    getMessageWindow().setVisible( false );
        getPanoramaPanel().setOwner(this);
2354    resetMessageArea();
        // the below code is inserted here due to difficulty using the visual composer to properly display
             the top and
2356    // bottom components of the outer split pane.
        getOuterSplitPane().add( getInnerSplitPane(), "bottom");
2358    center();
        getClientManager().configure();
2360

2362    // user code end

2364 }
     /**
2366  * Insert the method's description here.
      * Creation date: (18-06-00 17:16:36)
2368  * @param minimum int
      * @param maximum int
2370  * @param value int
      * @param string java.lang.String
2372  */
     public void initMessageArea(int minimum, int maximum, int value, String string)
```

```
{
   ivjMessageArea.setMinimum( minimum );
   ivjMessageArea.setMaximum( maximum );
   ivjMessageArea.setValue( value );
   ivjMessageArea.setString( string );
   getMessageWindow().addString( string );
   // make sure the message disappears after a set interval if the progress bar is fixed
   if ( minimum == maximum )
   {
      initMessageAreaTimer();
   }
   // turn off any ticking timers so that update is not interrupted
   else
   {
      if ( getMessageAreaTimer() != null )
      {
         getMessageAreaTimer().stop();
      }
   }
}
/**
 * Initialize the message area timer.
 * Creation date: (08-09-00 23:53:48)
 */
private void initMessageAreaTimer()
{
   // start a timer to allow the current message a brief display interval before being deleted.
   if ( ( getMessageAreaTimer() == null )||
    ( getMessageAreaTimer().isRunning() == false ))
   {
      setMessageAreaTimer( new Timer
      ( ClientConstants.getRESET_MESSAGE_AREA_DELAY(),
         new ActionListener()
         {
            public void actionPerformed( ActionEvent ae )
            {
               resetMessageArea();
            }
         }
      ) );
      getMessageAreaTimer().setRepeats( false );
      getMessageAreaTimer().start();
   }
}
/**
 * Return the PanoramaPanel property value.
 * @return clientapp.ClientPanorama
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public ClientPanorama initPanoramaPanel()
{
   if (ivjPanoramaPanel == null ){
      try {
         ivjPanoramaPanel = new clientapp.ClientPanorama();
         ivjPanoramaPanel.setName("PanoramaPanel");
         ivjPanoramaPanel.setPreferredSize(new java.awt.Dimension(300, 300));
         ivjPanoramaPanel.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
         ivjPanoramaPanel.setMinimumSize(new java.awt.Dimension(1, 1));
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
```

```
2440      return ivjPanoramaPanel;
     }
2442 /**
      * Insert the method's description here.
2444  * Creation date: (05-09-00 22:15:03)
      * @return boolean
2446  */
     public boolean isItemListenerActive() {
2448      return itemListenerActive;
     }
2450 /**
      * Insert the method's description here.
2452  * Creation date: (15-08-00 08:48:45)
      * @return boolean
2454  */
     public boolean isTerminated() {
2456      return terminated;
     }
2458 /**
      * Lock interface by disabling those menu items that give access to activities that run in separate
          threads.
2460  * The purpose of this is to minimize the potential for thread conflicts when operations such as
          adddocuments and
      * removedocuments are set to execute in their own threads.
2462  * Creation date: (09-10-00 21:49:32)
      * @param value boolean
2464  */
     public void lockInterface(boolean value)
2466 {
        getAddDocuments().setEnabled( !value );
2468     getRemoveDocuments().setEnabled( !value );
        getGroups().setEnabled( !value );
2470     getPanoramaPanel().setPanoramaLock( value );

2472 }
     /**
2474  * main entrypoint - starts the part when it is run as an application
      * @param args java.lang.String[]
2476  */
     public static void main(java.lang.String[] args)
2478 {
        try {
2480        ClientApplication aClientApplication;
            aClientApplication = new ClientApplication();
2482        aClientApplication.addWindowListener(new java.awt.event.WindowAdapter() {
               public void windowClosing(java.awt.event.WindowEvent e) {
2484              System.exit(0);
               };
2486        });
            aClientApplication.setVisible(true);
2488     } catch (Throwable exception) {
            System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JFrame");
2490        exception.printStackTrace(System.out);
        }
2492 }
     /**
2494  * PreferencesButtonToDialog: (PreferencesButton.action.actionPerformed(java.awt.event.ActionEvent) -->
          PreferencesDialog1.show()V)
      * @param arg1 java.awt.event.ActionEvent
2496  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
2498 private void PreferencesButtonToDialog(java.awt.event.ActionEvent arg1) {
        try {
2500        // user code begin {1}
            // user code end
2502        getPreferencesDialog1().show();
```

```
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM6: (PreferencesItem.action.actionPerformed(java.awt.event.ActionEvent) --> PreferencesDialog1
     .show()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void PreferencesMenuItemToDialog(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      getPreferencesDialog1().show();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Prepare termination of the object.
 * Creation date: (28-07-00 00:34:11)
 */
public void prepareTermination()
{
   if ( getClientManager().getClientInfo().isLocalMode() == false )
   {
      getClientManager().disconnectFromGroup( getClientManager().getClientInfo().getActiveGroup() );
   }
   getClientManager().informServerOfExit();
   getClientManager().getClientInfo().writePreferences();
   setTerminated( true );
}
/**
 * Remove a document from the visible document list box.
 * Creation date: (30-08-00 15:43:31)
 * @param senderName java.lang.String
 * @param documentName java.lang.String
 */
public void removeDocumentFromVisibleList(String senderName, String documentName)
{
   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
   if ( getVisibleDocumentsList().containsKey( senderName ))
   {
      ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
      for ( int i = 0; i < docList.size(); i++ )
      {
         if ( docList.get( i )instanceof DocumentItem )
         {
            if ( ( (DocumentItem) docList.get( i )).getDocumentID().equals( fqPath ))
            {
               docList.remove( i );
            }
         }
      }
      // if group member no longer contributes to panorama, i.e. if the author's name is the only entry
            ,then remove the list
      if ( docList.size() == 1 )
```

```
        {
2568       getVisibleDocumentsList().remove( senderName );
        }
2570    }
     }
2572 /**
      * Clear the message area of the client application main window, i.e. empty it so that it appears blank
          to the user.
2574  * Creation date: (18-06-00 17:17:37)
      */
2576 public void resetMessageArea()
     {
2578    getMessageArea().setSize( getMessagePanel().getSize() );
        getMessageArea().setString(ClientConstants.getMESSAGE_AREA_DEFAULT_STRING());
2580    getMessageArea().setValue(0);
        validate();
2582    repaint();
     }
2584 /**
      * Set method for the client mananager object.
2586  * Creation date: (16-06-00 00:04:31)
      * @param newClientManager clientapp.ClientManager
2588  */
     public void setClientManager(ClientManager newClientManager) {
2590    clientManager = newClientManager;
     }
2592 /**
      * Set method for the itemListenerActive field.
2594  * Creation date: (05-09-00 22:15:03)
      * @param newItemListenerActive boolean
2596  */
     public void setItemListenerActive(boolean newItemListenerActive) {
2598    itemListenerActive = newItemListenerActive;
     }
2600 /**
      * Set the message display area parameters to reflect the contents of the argument or reset to default
          values if argument is null.
2602  * Creation date: (10-07-00 14:44:52)
      * @param newMessage java.lang.String
2604  */
     public void setMessage( Message message )
2606 {
        if ( message != null )
2608    {
           getMessageDisplayArea().setText( message.getContents() );
2610       getMessageDisplayArea().setToolTipText( message.toString() );
        }
2612    else
        {
2614       getMessageDisplayArea().setText( "" );
           getMessageDisplayArea().setToolTipText( null );
2616    }
        getMessageDisplayArea().repaint();
2618
     }
2620 /**
      * Set method for the messageAreaTimer field.
2622  * Creation date: (31-08-00 02:18:56)
      * @param newMessageAreaTimer javax.swing.Timer
2624  */
     public void setMessageAreaTimer(javax.swing.Timer newMessageAreaTimer) {
2626    messageAreaTimer = newMessageAreaTimer;
     }
2628 /**
      * Set method for the messsageWindow field.
2630  * Creation date: (16-08-00 17:05:09)
```

```
     * @param newMessageWindow clientapp.MessageWindow
2632  */
     public void setMessageWindow(MessageWindow newMessageWindow) {
2634    messageWindow = newMessageWindow;
     }
2636 /**
     * Set method for the terminated field.
2638  * Creation date: (15-08-00 08:48:45)
     * @param newTerminated boolean
2640  */
     public void setTerminated(boolean newTerminated) {
2642    terminated = newTerminated;
     }
2644 /**
     * Set method for the discussion tree field.
2646  * Creation date: (09-07-00 23:03:10)
     * @param newTree DefaultTreeModel
2648  */
     public void setTopicTree(javax.swing.tree.DefaultTreeModel newTree)
2650 {
       // reset selection to top row to avoid triggering the change listener and have a spurious message
            inserted into the message area
2652   if ( getDiscussionTree().isSelectionEmpty() == false )
       {
2654     getDiscussionTree().setSelectionRow( 0 );
       }
2656   getDiscussionTree().setModel( newTree);
       getDiscussionTree().revalidate();
2658   getDiscussionTree().repaint();
       // System.out.println( "Inde i setTopicTree" );
2660 }
     /**
2662  * Set visible documents list.
     * Creation date: (30-08-00 16:14:33)
2664  * @param newVisibleDocumentsList java.util.Hashtable
     */
2666 public void setVisibleDocumentsList(java.util.Hashtable newVisibleDocumentsList)
     {
2668   visibleDocumentsList = newVisibleDocumentsList;
     }
2670 /**
     * Initialize the message properties box with the contents of the current message, if any.
2672  * Creation date: (19-07-00 20:11:38)
     */
2674 public void showMessageProperties()
     {
2676   if ( getDiscussionTree().getSelectionCount() == 0 )
       {
2678     ErrorBox noSelectionErrorBox = new ErrorBox();
         noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
2680     noSelectionErrorBox.show();
         return;
2682   }
       else if ( getClientManager().verifyGroupConnectionManagement() == false )
2684   {
         return;
2686   }
       else if ( getClientManager().getClientInfo().getCurrentMessage() != null )
2688   {
         MessagePropertiesBox messagePropertiesBox = new MessagePropertiesBox();
2690     messagePropertiesBox.initTable( getClientManager().getClientInfo().getCurrentMessage() );
         messagePropertiesBox.show();
2692   }
     }
2694 /**
```

```
     * The message window is the log window that can be activate by double clicking the message bar at the
         bottom of the
2696  * client applicaton window. This method makes that message window visible.
     * Creation date: (16-08-00 17:43:02)
2698  */
     public void showMessageWindow()
2700 {
        getMessageWindow().setVisible( true );
2702 }
     /**
2704  * Prepare termination of the client application and then dispose of it.
     * Creation date: (12-09-00 21:19:40)
2706  */
     public void terminateAndClose()
2708 {
        prepareTermination();
2710    getClientManager().prepareForTermination();
        this.dispose();
2712 }
     /**
2714  * Test method. Purely for development purposes. Will be removed in stable releases.
     * Creation date: (29-06-00 18:52:19)
2716  */
     public void test()
2718 {
        PeerViewClient client = new PeerViewClient("PeerViewClient");
2720 // DefaultClient client = new DefaultClient("WTEST");

2722    try
        {
2724       URLString url = URLString.createSessionURL("localhost", 4461 ,"socket","test");
           if (SessionFactory.sessionExists( url ))
2726       {
              // System.out.println( "Session findes" );
2728       }
           Session session = SessionFactory.createSession(client, url, true );
2730       if ( session.channelExists( "test" ))
           {
2732          // System.out.println( "channelExists" );
              Channel channel = session.createChannel(client, "test", true, true, true);
2734
              channel.addConsumer(client,client);
2736       }
        }
2738    catch (JSDTException JSDTE)
        {
2740       getClientManager().handleJSDTException( JSDTE );
        }
2742
     }
2744 /**
     * Change value field of the progress/message bar at the bottom of the client application main window.
2746  * Creation date: (18-06-00 17:17:07)
     * @param newValue int
2748  */
     public void updateMessageArea(int newValue)
2750 {
        ivjMessageArea.getModel().setValue(newValue);
2752    ivjMessageArea.repaint();
     // ivjMessageArea.paintImmediately(ivjMessageArea.getBounds());
2754 }
     /**
2756  * Convenience method for updating the visible documents box.
     * Creation date: (30-08-00 22:30:55)
2758  */
     public void updateVisibleDocumentsBox()
```

```
{
   updateVisibleDocumentsBox( getVisibleDocumentsList() );
}
/**
 * Update contents of visible documents box.
 * Creation date: (30-08-00 01:03:02)
 * @param documents java.util.Hashtable
 */
public void updateVisibleDocumentsBox(Hashtable documents)
{
   // temporarily disable itemlistener to avoid repeated triggering
   // the below ought to be substitable with a call to removeAllItem but an attempt to do so seemed to
        fail.
   setItemListenerActive( false );
/* while ( getDocumentBox().getItemCount() > 0 )
   {
      getDocumentBox().removeItemAt( 0 );
   }
*/
   getDocumentBox().setModel( new DefaultComboBoxModel() );
   getDocumentBox().insertItemAt( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION() )
        , 0 );
   Collection values = documents.values();
   Iterator iterator = values.iterator();
   while ( iterator.hasNext() )
   {
      ArrayList itemList = (ArrayList) iterator.next();
      Iterator itemListIterator = itemList.iterator();
      while ( itemListIterator.hasNext() )
      {
         Object item = itemListIterator.next();
         getDocumentBox().addItem( item );
         // System.out.println( item.toString() );
      }
   }
   setItemListenerActive( true );
   getDocumentBox().setSelectedIndex( 0 );
}
}
```

Listing C.4: ClientInfo.java

```
package clientapp;

/**
 *
 * Creation date: (02-07-00 18:41:23)
 * @author:
 */
 import java.util.Hashtable;
 import java.util.Properties;
 import java.io.File;
 import java.util.HashSet;
 import java.io.*;
 import java.awt.datatransfer.*;

 /** The main entity class for the client application. This class holds data structures and fields that
      must be shared by
 * several client application components.
 */
public class ClientInfo implements ClipboardOwner
{
   private com.sun.media.jsdt.Session groupManagementSession = null;
   private static java.util.Properties preferences = null;
   private java.util.Hashtable groups = null;
   private com.sun.media.jsdt.Channel groupManagementChannel = null;
   private peerviewmisc.WorkGroup activeGroup = null;
```

```
    private java.lang.String currentDocumentName = null;
    private peerviewmisc.Message currentMessage = null;
    private java.util.HashSet localFiles = new HashSet();
    private java.awt.datatransfer.Clipboard clipBoard = new Clipboard( ClientConstants.getCLIPBOARD_NAME
        () );
    private boolean localMode = true;
/**
 * ClientInfo constructor comment.
 */
public ClientInfo() {
    super();
    initialize();
}
/**
 * Create unique client id as concatenation of default name + system time in milliseconds + local host
     IP address or
 * default name + system time if the IP address is unavailable.
 * Creation date: (16-07-00 13:50:53)
 * @return java.lang.String
 */
public static String createUniqueClientID()
{
    String ID = ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )+ (new java.util.Date
        ().toString() );
    try
    {
    ID = ID + java.net.InetAddress.getLocalHost().toString();
    }
    catch ( java.net.UnknownHostException UHE )
    {
        System.err.println( UHE.toString() );
        System.exit( 0 );
    }
    return ID;
}
/**
 * Insert the method's description here.
 * Creation date: (16-07-00 13:38:00)
 * @exception java.lang.Throwable The exception description.
 */
protected void finalize() throws java.lang.Throwable
{
    // write preferences
}
/**
 * Insert the method's description here.
 * Creation date: (06-07-00 13:15:33)
 * @return peerviewmisc.WorkGroup
 */
public peerviewmisc.WorkGroup getActiveGroup() {
    return activeGroup;
}
/**
 * Insert the method's description here.
 * Creation date: (07-08-00 15:57:43)
 * @return java.awt.datatransfer.Clipboard
 */
public java.awt.datatransfer.Clipboard getClipBoard() {
    return clipBoard;
}
/**
 * Insert the method's description here.
 * Creation date: (11-07-00 15:18:55)
 * @return java.lang.String
 */
public java.lang.String getCurrentDocumentName() {
```

```
88      return currentDocumentName;
    }
90  /**
     * Insert the method's description here.
92   * Creation date: (19-07-00 17:30:43)
     * @return peerviewmisc.Message
94   */
    public peerviewmisc.Message getCurrentMessage() {
96      return currentMessage;
    }
98  /**
     * Insert the method's description here.
100  * Creation date: (04-07-00 17:42:59)
     * @return com.sun.media.jsdt.Channel
102  */
    public com.sun.media.jsdt.Channel getGroupManagementChannel() {
104     return groupManagementChannel;
    }
106 /**
     * Insert the method's description here.
108  * Creation date: (02-07-00 18:57:40)
     * @return com.sun.media.jsdt.Session
110  */
    public com.sun.media.jsdt.Session getGroupManagementSession() {
112     return groupManagementSession;
    }
114 /**
     * Insert the method's description here.
116  * Creation date: (04-07-00 16:21:40)
     * @return java.util.Hashtable
118  */
    public java.util.Hashtable getGroups() {
120     return groups;
    }
122 /**
     * Insert the method's description here.
124  * Creation date: (03-08-00 01:49:44)
     * @return java.util.HashSet
126  */
    public java.util.HashSet getLocalFiles() {
128     return localFiles;
    }
130 /**
     * Insert the method's description here.
132  * Creation date: (02-07-00 22:08:51)
     * @return java.util.Properties
134  */
    public static java.util.Properties getPreferences()
136 {
        if ( preferences == null)
138     {
           preferences = new Properties( ClientConstants.getDEFAULT_PREFERENCES() );
140        readPreferences();
           // if the client has not yet been assigned a unique identifier, do so now. The new ID will be
                saved to disk upon exit
142        if ( ( (String) ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )).
                equalsIgnoreCase( ClientConstants.getDEFAULT_CLIENT_NAME() ))
           {
144           ClientInfo.getPreferences().put( ClientConstants.getNAME(), createUniqueClientID() );
              // System.out.println( ClientInfo.getPreferences().get( ClientConstants.getNAME() ));
146        }
        }
148     return preferences;
    }
150 /**
     * Insert the method's description here.
```

```
 * Creation date: (16-07-00 13:36:57)
 */
public void initialize()
{
}
/**
 * Insert the method's description here.
 * Creation date: (10-08-00 16:37:24)
 * @return boolean
 */
public boolean isLocalMode() {
   return localMode;
}
/**
 * Obligatory method for meeting the ClipboardOwner interface specification
 * Creation date: (07-08-00 17:28:28)
 * @param clipBoard java.awt.datatransfer.Clipboard
 * @param contents java.awt.datatransfer.Transferable
 */
public void lostOwnership(Clipboard clipBoard, Transferable contents)
{
}
/**
 * Read preferences from Properties object on disk. Each call to this function results in a read
 * from disk rather than an internal lookup, i.e. the values read from disk are not cached as a class
     member.
 * This eliminates data redundancy and ensures consistency. The overhead is neglible as the file in
     question is essentially
 * plain text.
 * Algorithm:
 * 1. Construct new Properties object P
 * 2. Initialize P with default preferences
 * 3. Read user preferences from disk file
 * 4. IF read is successful THEN
 * 4.1. Store user preferences in P
 * 5. Return P
 * This approach will return a valid object even if the read fails or in case of the application being
     used for the first
 * time.
 * Creation date: (25-06-00 13:54:55)
 */
public static void readPreferences()
{
   try
   {
      FileInputStream f = new FileInputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
      preferences.load(f);
      f.close();
   }
   catch (FileNotFoundException ffe)
   {
      // the exception can be handled by ignoring it since the default values in p will substitute for
          what should
      // have been read from file.
   }
   catch (IOException ie)
   {
      // any IOException != FileNotFound should be dealt with in the conventional manner
      System.err.println( ie.toString() );
      System.exit(0);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (06-07-00 13:15:33)
 * @param newActiveGroup peerviewmisc.WorkGroup
```

```
214   */
      public void setActiveGroup(peerviewmisc.WorkGroup newActiveGroup) {
216     activeGroup = newActiveGroup;
      }
218   /**
       * Insert the method's description here.
220    * Creation date: (07-08-00 15:57:43)
       * @param newClipBoard java.awt.datatransfer.Clipboard
222    */
      public void setClipBoard(java.awt.datatransfer.Clipboard newClipBoard) {
224     clipBoard = newClipBoard;
      }
226   /**
       * Insert the method's description here.
228    * Creation date: (11-07-00 15:18:55)
       * @param newCurrentDocumentName java.lang.String
230    */
      public void setCurrentDocumentName(java.lang.String newCurrentDocumentName) {
232     currentDocumentName = newCurrentDocumentName;
      }
234   /**
       * Insert the method's description here.
236    * Creation date: (19-07-00 17:30:43)
       * @param newCurrentMessage peerviewmisc.Message
238    */
      public void setCurrentMessage(peerviewmisc.Message newCurrentMessage) {
240     currentMessage = newCurrentMessage;
      }
242   /**
       * Insert the method's description here.
244    * Creation date: (04-07-00 17:42:59)
       * @param newGroupManagementChannel com.sun.media.jsdt.Channel
246    */
      public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
248     groupManagementChannel = newGroupManagementChannel;
      }
250   /**
       * Insert the method's description here.
252    * Creation date: (02-07-00 18:57:40)
       * @param newGroupManagementSession com.sun.media.jsdt.Session
254    */
      public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
256     groupManagementSession = newGroupManagementSession;
      }
258   /**
       * Insert the method's description here.
260    * Creation date: (04-07-00 16:21:40)
       * @param newGroups java.util.Hashtable
262    */
      public void setGroups(java.util.Hashtable newGroups) {
264     groups = newGroups;
      }
266   /**
       * Insert the method's description here.
268    * Creation date: (03-08-00 01:49:44)
       * @param newLocalFiles java.util.HashSet
270    */
      public void setLocalFiles(java.util.HashSet newLocalFiles) {
272     localFiles = newLocalFiles;
      }
274   /**
       * Insert the method's description here.
276    * Creation date: (10-08-00 16:37:24)
       * @param newLocalMode boolean
278    */
      public void setLocalMode(boolean newLocalMode) {
```

```
    localMode = newLocalMode;
  }
  /**
   * Insert the method's description here.
   * Creation date: (02-07-00 22:08:51)
   * @param newPreferences java.util.Properties
   */
  public static void setPreferences(java.util.Properties newPreferences) {
    preferences = newPreferences;
  }
  /**
   * Write preferences to disk file.
   * Creation date: (17-07-00 12:13:32)
   * @param preferences java.util.Properties
   */
  public static void writePreferences()
  {
    // write Properties object to disk
    try
    {
      FileOutputStream f = new FileOutputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
      preferences.store( f, ClientConstants.getPROPERTIES_HEADER() );
      f.close();
    }
    catch (IOException ie)
    {
      System.err.println( ie.toString() );
      System.exit(0);
    }

  }
}
```

Listing C.5: ClientManager.java

```
package clientapp;


 import java.io.*;
 import javax.swing.JComponent;
 import javax.swing.*;
 import javax.swing.ImageIcon;
 import javax.swing.JEditorPane;
 import javax.swing.JTextArea;
 import com.sun.media.jsdt.*;
 import com.sun.media.jsdt.event.*;
 import java.util.Properties;
 import java.util.Hashtable;
 import peerviewmisc.*;
 import java.util.zip.*;
 import java.util.Enumeration;
 import java.util.Iterator;
 import java.util.Random;
 import javax.swing.tree.*;
 import javax.swing.event.*;
 import java.util.GregorianCalendar;
 import java.awt.datatransfer.*;

/**
 * The main control class for the client application. This class holds methods and control structures
     shared by several
 * client application components.
 * Creation date: (14-06-00 21:27:43)
 * @author:
 */
public class ClientManager
{
```

```
/* public class DocPackage implements Serializable
   {
      String docAndPathName;
      java.awt.Rectangle bounds;

      public DocPackage( String docAndPathNameArg, java.awt.Rectangle boundsArg )
      {
         docAndPathName = docAndPathNameArg;
         bounds = boundsArg;
      }

      public String getDocAndPathName()
      {
         return docAndPathName;
      }

      public java.awt.Rectangle getBounds()
      {
         return bounds;
      }

      public void setDocAndPathName( String docAndPathNameArg )
      {
         docAndPathName = docAndPathNameArg;
      }

      public void setBounds( java.awt.Rectangle boundsArg )
      {
         bounds = boundsArg;
      }
   };
*/

   /** Event handler for JSDT connections. Ensures automatic notification in case of connection failure
   */
   public class GroupManagementListener implements ConnectionListener
   {
      public void connectionFailed( ConnectionEvent ce )
      {
         groupManagementConnectionFailed( ce );
      }
   };

   // the below inner classes derived from Thread are used instead of member methods to allow execution
        in separate threads.
   // This is necessary to allow the event-dispatching thread to process events and thus enable proper
        updating of the message area progress bar
   // in the client main window. Initially, I used paintImmediately to achieve the desired effect but
        that seemed to produce flicker on some platforms and
   // so had to be abandoned.
   /** An object of this class should be instantiated with a set of documents and launched in a separate
         thread so that
   *  it executes without blocking the event dispatching thread and thus the GUI.
   */
   public class RemoveDistributedDocumentsFromPanorama extends Thread
   {
      public void run()
      {
         owner.lockInterface( true );
         java.util.Set keys = owner.getPanoramaPanel().getDocuments().keySet();
         Iterator iterator = keys.iterator();
         owner.initMessageArea( 1, owner.getPanoramaPanel().getDocuments().size(), 1, ClientConstants.
              getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() );
         // copy to array first to avoid concurrentModificationException
         // this should be considered an interim solution till I come up with a cleaner alternative
         String keyArray[] = new String[ keys.size() ];
```

```
      for ( int i = 0 ; iterator.hasNext(); i++ )
      {
         keyArray[i] = (String) iterator.next();
      }
      for ( int i = 0; i < keyArray.length; i++ )
      {
         String key = keyArray[i];
         edu.umd.cs.jazz.ZVisualLeaf leaf = (edu.umd.cs.jazz.ZVisualLeaf) owner.getPanoramaPanel().
             getDocuments().get( key );
         owner.getPanoramaPanel().removeGlobalDocument( (String) leaf.getClientProperty(
             ClientConstants.getSENDER_NAME() ),
                                        (String) leaf.getClientProperty( ClientConstants.
                                            getDOCUMENT_NAME() ));
         owner.advanceTask();
         yield();
      }
      owner.getPanoramaPanel().getDocuments().clear();
      owner.resetMessageArea();
      owner.lockInterface( false );
   }
};
/** An object of this class should be instantiated with a set S of file names and launched in a
     separate thread. It will
* add the files whose names are in S to the panorama.
*/
public class AddDocuments extends Thread
{
   File newFiles[];

   public AddDocuments( File newFilesArg[] )
   {
      newFiles = newFilesArg;
   }

   public void run()
   {
      owner.lockInterface( true );
      boolean duplicateFlag = false;
      // prepare progress indicator for the document retrieval process
      owner.initMessageArea( 0, newFiles.length, 0, ClientConstants.getRETRIEVING_DOCUMENTS() );
      java.util.HashSet fileSet = new java.util.HashSet();
      for (int i = 0; i < newFiles.length; i++)
      {
         owner.advanceTask();
         if ( getClientInfo().getLocalFiles().contains( newFiles[i] )== true )
         {
            // if file is already in the panorama, raise the flag and continue with the next file in
                the array
            duplicateFlag = true;
            continue;
         }
         else
         {
            getClientInfo().getLocalFiles().add( newFiles[i] );
         }
         try
         {
         owner.getPanoramaPanel().addLocalDocument(  readDocument( newFiles[i] ),
                                    ClientInfo.getPreferences().getProperty( ClientConstants.
                                        getNAME() ),
                                    ClientInfo.getPreferences().getProperty( ClientConstants.
                                        getAUTHOR_NAME() ),
                                    newFiles[i].getPath(),
                                    newFiles[i].lastModified()
                                         );
         }
```

```
152            catch ( IOException IOE )
               {
154
               }
156         informServerOfNewDocument( newFiles[i].getPath() );
            fileSet.add( newFiles[i] );
158         yield();
            }
160         broadcastDocuments( fileSet );
            updatePanorama();
162         // System.out.println("inde i addDocuments åtrd");
            owner.resetMessageArea();
164         if ( duplicateFlag == true )
            {
166            MessageBox messageBox = new MessageBox();
               messageBox.setText( ClientConstants.getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() );
168            messageBox.show();
            }
170         owner.lockInterface( false );
         }
172   };
      /** This class should be instantiated with a set S of files and launched in a separate thread. It
           will remove
174   * from the panorama the documents corresponding the members of S.
      */
176   public class RemoveDocuments extends Thread
      {
178      File[] removables;

180      public RemoveDocuments( File[] removablesArg )
         {
182         removables = removablesArg;
         }
184
         public void run()
186      {
            owner.lockInterface( true );
188         owner.getPanoramaPanel().setPanoramaLock( true);
            owner.initMessageArea(1, removables.length, 1, ClientConstants.getREMOVING_DOCUMENTS() );
190         java.util.Vector filePaths = new java.util.Vector();
            for (int i = 0 ; i < removables.length; i++)
192         {  owner.getPanoramaPanel().removeLocalDocument( ClientInfo.getPreferences().getProperty(
                 ClientConstants.getNAME() ), removables[i].getPath() );
               getClientInfo().getLocalFiles().remove( removables[i] );
194            filePaths.add( removables[i].getPath() );
               owner.advanceTask();
196            yield();
            }
198         if ( getClientInfo().isLocalMode() == false )
            {
200            broadcastDocumentRemovals( filePaths );
            }
202         updatePanorama();
            owner.resetMessageArea();
204         owner.getPanoramaPanel().setPanoramaLock( false );
            owner.lockInterface( false );
206      }
      };
208   /** This class should be instantiated with a recipient ID and author name and launched in a separate
           thread. It
      * will send to the recipient the set of local documents currently displayed in the panorama.
210   */
      public class UnicastDocuments extends Thread
212   {
         String recipientName;
214      String recipientAuthorName;
```

```
216     public UnicastDocuments( String recipientNameArg, String recipientAuthorNameArg )
        {
218        recipientName = recipientNameArg;
           recipientAuthorName = recipientAuthorNameArg;
220     }

222     public void run()
        {
224        if ( getClientInfo().getLocalFiles().size() > 0 )
           {
226           WorkGroup activeGroup = getClientInfo().getActiveGroup();
              URLString url = null;
228           Session session = null;
              Channel groupChannel = null;
230           java.util.Vector rawDocumentData = new java.util.Vector();
                 try
232           {
                 url = activeGroup.getSessionURL();
234              session = SessionFactory.createSession( getGroupManagementClient(), url, false );
                 groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
                      getWorkGroupChannelID(), true, true, false);
236           }
                 catch ( JSDTException JSDTE )
238           {
                 handleJSDTException( JSDTE );
240              return;
              }
242           owner.initMessageArea(1, getClientInfo().getLocalFiles().size(), 1,
                               ClientConstants.getUNICASTING_DOCUMENTS_MESSAGE( getClientInfo().
                                    getLocalFiles().size(),
244                         recipientAuthorName ));
              Iterator iterator = getClientInfo().getLocalFiles().iterator();
246           while ( iterator.hasNext() )
              {
248              File file = (File)iterator.next();
                 // read contents of file and compress it
250              try
                 {
252                 byte fileContents[] = readDocument( file );
                    CompressedDocumentPackage compressedDocumentPackage = deflateDocument( fileContents );
254                 compressedDocumentPackage.setPath( file.getPath() );
                    // separate name from path here to avoid confusion over separator char across platforms
                         .
256                 compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf
                         ( File.separator )+ 1 ));
                    rawDocumentData.add( compressedDocumentPackage );
258                 owner.advanceTask();
                    yield();
260              }
                 catch ( IOException IOE )
262              {
                    System.err.println( IOE.toString() );
264                 System.exit( 0 );
                 }
266           }
              try
268           {
                 DataPackage outgoingPackage = new DataPackage();
270              outgoingPackage.setContents( rawDocumentData );
                 outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants
                      .getNEW_DOCUMENTS() );
272              outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), getClientInfo()
                      .getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ));
                 Data data = new Data( outgoingPackage );
274              groupChannel.sendToClient( getGroupManagementClient(), recipientName, data );
```

```
            }
276         catch ( JSDTException JSDTE )
            {
278            handleJSDTException( JSDTE );
            }
280         owner.resetMessageArea();
         }
282      }
      };
284
      /** Use this class to remove non-local documents from the panorama.
286   * This class should be instantiated with the name of an author (to be included in the notification
          message shown
      * to the user), a sender (which will be used to identify the documents) and a set of file names.
          Objects should be
288   * launched in a separate thread.
      */
290   public class RemoveDistributedDocuments extends Thread
      {
292      String authorName;
         String senderName;
294      java.util.Vector filePaths;

296      public RemoveDistributedDocuments(String authorNameArg, String senderNameArg, java.util.Vector
             filePathsArg )
         {
298         authorName = authorNameArg;
            senderName = senderNameArg;
300         filePaths = filePathsArg;
         }
302
         public void run()
304      {
            owner.lockInterface( true );
306         owner.initMessageArea( 1, filePaths.size(), 1, ClientConstants.
                getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST( authorName, filePaths.size() ));
            Iterator iterator = filePaths.iterator();
308         while ( iterator.hasNext() )
            {
310            owner.getPanoramaPanel().removeGlobalDocument( senderName, (String) iterator.next() );
               owner.advanceTask();
312            yield();
            }
314         updatePanorama();
            owner.resetMessageArea();
316         owner.lockInterface( false );
         }
318   };
      /** This class should be used to add non-local documents to the panorama. It accepts a documents
          package sent by
320   * another client and the ID of that client. Objects of this class should be launched in a separate
          thread.
      */
322   public class AddDistributedDocuments extends Thread
      {
324      DataPackage incomingPackage = null;
         String senderName = null;
326      public AddDistributedDocuments( DataPackage incomingPackageArg, String senderNameArg )
         {
328         incomingPackage = incomingPackageArg;
            senderName = senderNameArg;
330      }

332      public void run()
         {
334         owner.lockInterface( true );
```

```
         java.util.Vector documents = (java.util.Vector) incomingPackage.getContents();
         String authorName = incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() );
         owner.initMessageArea( 1, documents.size(), 1, ClientConstants.getADDING_DISTRIBUTED_DOCUMENTS(
             authorName, documents.size() ));
         for ( int i = 0 ; i < documents.size(); i++ )
         {
            CompressedDocumentPackage compressedDocumentPackage = (CompressedDocumentPackage) documents.
                get( i );
            byte[] rawDocumentData = inflateDocument( compressedDocumentPackage );
            owner.getPanoramaPanel().addGlobalDocument( rawDocumentData, senderName, authorName,
                compressedDocumentPackage.getPath(), compressedDocumentPackage.getName() );
            owner.advanceTask();
            yield();
         }
         updatePanorama();
         owner.resetMessageArea();
         owner.lockInterface( false );
      }
   };

   /** Client object to be used with the group management session.
   */
   public class GroupManagementClient implements Client, ChannelConsumer
   {
      String name;

      public GroupManagementClient()
      {
      }

      public String getName()
      {
         return name;
      }

      public void setName( String nameArg )
      {
         name = nameArg;
      }

      /** Accept and parse incoming data. The if-else structure maps message types to functions.
      */
      public synchronized void dataReceived( Data data )
      {
         DataPackage incomingPackage = null;
         try
         {
            incomingPackage = (DataPackage) data.getDataAsObject();
         }
         catch ( Exception E )
         {
            System.err.println( E.toString() );
            System.exit(0);
         }
         if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
              equalsIgnoreCase ( ClientConstants.getGROUP_DIRECTORY() ))
         {
            updateGroupDirectory( (Hashtable) incomingPackage.getContents() );
         }
         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
              equalsIgnoreCase ( ClientConstants.getNEW_DOCUMENT() ))
         {
            addDistributedDocument( incomingPackage, data.getSenderName() );
         }
         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
              equalsIgnoreCase ( ClientConstants.getNEW_DOCUMENTS() ))
```

```
        {
396        (new AddDistributedDocuments( incomingPackage, data.getSenderName() )).start();
        }
398     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase ( ClientConstants.getREQUEST_DOCUMENTS() ))
        {
400        (new UnicastDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty(
               DataPackage.getAUTHOR_NAME() ))).start();
        }
402
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase ( ClientConstants.getTOPIC_TREE() ))
404     {
           setTopicTree( (DefaultTreeModel) incomingPackage.getContents(), incomingPackage.getProperties
               ().getProperty( DataPackage.getDOCPATH() ));
406     }

408     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getMESSAGE() ))
        {
410        setDisplayedMessage( (Message) incomingPackage.getContents() );
        }
412
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getREMOVE_DOCUMENT() ))
414     {
           removeDistributedDocument( incomingPackage.getProperties().getProperty( DataPackage.
               getAUTHOR_NAME() ), data.getSenderName(), (String) incomingPackage.getContents() );
416     }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getREMOVE_DOCUMENTS() ))
418     {
           (new RemoveDistributedDocuments( incomingPackage.getProperties().getProperty( DataPackage.
               getAUTHOR_NAME() ), incomingPackage.getProperties().getProperty( DataPackage.
               getCLIENT_NAME() ), (java.util.Vector) incomingPackage.getContents() )).start();
420     }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getCLIENT_LEFT_GROUP() ))
422     {
           owner.initMessageArea( 1, 1, 1, ClientConstants.getCLIENT_LEFT_GROUP_MESSAGE( (String)
               incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() )));
424     }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getDOCUMENT_UPDATE() ))
426     {
           updateDistributedDocument( incomingPackage.getProperties().getProperty( DataPackage.
               getDOCPATH() ), data.getSenderName(), (CompressedDocumentPackage) incomingPackage.
               getContents() );
428     }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getTOPIC_TREE_UPDATE() ))
430     {
           updateTopicTree( data.getSenderName(), (String) incomingPackage.getProperties().getProperty(
               DataPackage.getDOCPATH() ), (DefaultTreeModel) incomingPackage.getContents() );
432     }
        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
            equalsIgnoreCase( ClientConstants.getUPDATE_PANORAMA() ))
434     {
           updatePanoramaByRequest();
436     }
     }
438
     public Object authenticate ( AuthenticationInfo ai )
440  {
        return null;
442  }
```

```
   };

   private ClientApplication owner;
   private ClientInfo clientInfo = null;
   private GroupManagementClient groupManagementClient = null;
/**
 * ClientManager constructor comment.
 */
public ClientManager(ClientApplication newOwner)
{
   super();
   owner = newOwner;
   initialize();
}
/**
 * Inflate a data package received over a network connection and add it to the panorama using the data
      package properties
 as descriptive parameters.
 * Creation date: (08-07-00 18:08:33)
 * @param contents byte[]
 * @param path java.lang.String[]
 */
public void addDistributedDocument(DataPackage incomingPackage, String sender)
{
   byte[] rawFileContents = inflateDocument( (CompressedDocumentPackage) incomingPackage.getContents() )
        ;
   owner.getPanoramaPanel().addGlobalDocument( rawFileContents,
      sender,
      incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() ),
      incomingPackage.getProperties().getProperty( DataPackage.getDOCPATH() ),
      incomingPackage.getProperties().getProperty( DataPackage.getDOCNAME() ));
   owner.initMessageArea( 1, 1, 1, ClientConstants.getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(
      incomingPackage.getProperties().getProperty( DataPackage.getDOCPATH() ), incomingPackage.
          getProperties().getProperty( DataPackage.getAUTHOR_NAME() )));
}
/**
 * Add documents to the panorama by launching a separate thread to handle the task.
 * Creation date: (03-10-00 11:17:31)
 * @param newFiles java.io.File[]
 */
public void addDocuments(File[] newFiles)
{
   AddDocuments addDocuments = new AddDocuments( newFiles );
   addDocuments.start();
/* try
   {
      addDocuments.join();
   }
   catch ( InterruptedException ie )
   {
      System.err.println( ie.toString() );
   }
*/
}
/**
 * Read a listing of document names from disk and add the corresponding documents to the panorama.
 * Creation date: (11-10-00 13:19:00)
 * @param docs java.util.Vector
 */
public void addDocumentsFromDisk(java.util.Vector docs)
{
   if ( docs != null )
   {
      File files[] = new File[ docs.size() ];
      for ( int i = 0 ; i < docs.size(); i++ )
      {
```

```
506         DocPackage docPackage = (DocPackage) docs.get( i );
            File file = new File( docPackage.getDocNameAndPath() );
508         files[ i ] = file;
        }
510     AddDocuments addDocuments = new AddDocuments( files );
        addDocuments.start();
512     try
        {
514        addDocuments.join();
        }
516     catch ( InterruptedException ie )
        {
518        System.err.println( ie.toString() );
        }
520     // the below assumes that the localdocuments set now contains all the newly added documents from
            the above call to
        // addDocuments. A cleaner solution would be to merge the adding of files with the repositioning
            but the
522     // readDocumentList feature wasn't included in the original design, so pending a revision of the
            code, this should do OK.
        for ( int i = 0 ; i < docs.size(); i++ )
524     {
           DocPackage docPackage = (DocPackage) docs.get( i );
526        String fqName = ClientConstants.createFullyQualifiedDocumentName( ClientInfo.getPreferences().
               getProperty( ClientConstants.getNAME() ),
                                                        docPackage.getDocNameAndPath() );
528        edu.umd.cs.jazz.ZVisualLeaf leafToBeRepositioned = (edu.umd.cs.jazz.ZVisualLeaf) owner.
               getPanoramaPanel().getLocalDocuments().get( fqName );
           ( (PZSwing) leafToBeRepositioned.getVisualComponent() ).getComponent().setSize( docPackage.
               getSize() );
530        leafToBeRepositioned.editor().getTransformGroup().setTranslation(  (float) docPackage.
               getPosition().getWidth(),
                                                         (float) docPackage.getPosition().getHeight() );
532        leafToBeRepositioned.putClientProperty( ClientConstants.getNODE_LOCKED(), docPackage.
               getNodeLocked() );
        }
534     owner.getPanoramaPanel().validate();
        owner.getPanoramaPanel().repaint();
536  }
     }
538  /**
      * Send of list of files to be removed from the panorama to all members of the group this client is
          joined to, if any.
540   * Creation date: (26-07-00 02:30:02)
      * @param files java.io.File[]
542   */
     public void broadcastDocumentRemovals(java.util.Vector files)
544  {
        Session session = null;
546     Channel groupChannel = null;
        if (  ( getClientInfo().isLocalMode() == false )&&
548          ( files.size() > 0 ))
        {
550        WorkGroup activeGroup = getClientInfo().getActiveGroup();
           owner.initMessageArea(1, 1, 1, ClientConstants.getBROADCASTING_DOCUMENT_REMOVALS() );
552        DataPackage outgoingPackage = new DataPackage();
           outgoingPackage.setContents( files );
554        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
               getREMOVE_DOCUMENTS () );
           outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), activeGroup.
               getWorkGroupID() );
556        outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(),
                                        ClientInfo.getPreferences().getProperty( ClientConstants.
                                            getAUTHOR_NAME() ));
558        outgoingPackage.getProperties().setProperty( DataPackage.getCLIENT_NAME(),
```

```
                                        ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )
                                            );
560       try
          {
562          URLString url = getClientInfo().getActiveGroup().getSessionURL();
             session = SessionFactory.createSession( getGroupManagementClient(), url, false );
564          groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
                 getWorkGroupChannelID(), true, true, false);
             Data data = new Data( outgoingPackage );
566          groupChannel.sendToOthers( getGroupManagementClient(), data );
          }
568       catch ( JSDTException JSDTE )
          {
570          handleJSDTException( JSDTE );
          }
572       owner.resetMessageArea();
       }
574
    }
576 /**
     * Transmit a set of files to all members of the group that this client belongs to, if any.
578  * Creation date: (06-07-00 14:06:11)
     */
580 public void broadcastDocuments( java.util.HashSet files)
    {
582    if ( ( getClientInfo().isLocalMode() == false )&& ( files.size() > 0 ))
       {
584       WorkGroup activeGroup = getClientInfo().getActiveGroup();
          URLString url = null;
586       Session session = null;
          Channel groupChannel = null;
588       String clientNames[] = null;
          int numClientNames = 0;
590       java.util.Vector rawDocumentData = new java.util.Vector();
          try
592       {
             url = getClientInfo().getActiveGroup().getSessionURL();
594          session = SessionFactory.createSession( getGroupManagementClient(), url, false );
             groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
                 getWorkGroupChannelID(), true, true, false);
596       }
          catch ( JSDTException JSDTE )
598       {
             handleJSDTException( JSDTE );
600       }
          try
602       {
          numClientNames = groupChannel.listClientNames().length;
604       }
          catch ( JSDTException JSDTE )
606       {
             handleJSDTException( JSDTE );
608       }
          // inform user that broadcast is in progress
610       if ( numClientNames > 1 )
          {
612          owner.initMessageArea(1, files.size(), 1, ClientConstants.getBROADCASTING_DOCUMENTS_MESSAGE(
                 files.size() ));
          }
614       Iterator iterator = files.iterator();
          while ( iterator.hasNext() )
616       {
             File file = (File) iterator.next();
618          // if more members than this client
             CompressedDocumentPackage compressedDocumentPackage = null;
620          try
```

```
            {
622         // read contents of file and compress it
            byte fileContents[] = readDocument( file );
624         compressedDocumentPackage = deflateDocument( fileContents );
            compressedDocumentPackage.setPath( file.getPath() );
626         // separate name from path here to avoid confusion over separator char across platforms.
            compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf( File.
                separator )+ 1 ));
628         //System.out.println( "compressedPackageName" + compressedDocumentPackage.getName() );
            }
630         catch ( IOException IOE )
            {
632            System.err.println( IOE.toString() );
               System.exit( 0 );
634         }
            rawDocumentData.add( compressedDocumentPackage );
636         if ( numClientNames > 1 )
            {
638            owner.advanceTask();
            }
640      }
         // submit raw document data
642      if ( numClientNames > 1 )
         {
644         DataPackage outgoingPackage = new DataPackage();
            outgoingPackage.setContents( rawDocumentData );
646         outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
                getNEW_DOCUMENTS() );
            outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(),
648                                   ClientInfo.getPreferences().getProperty( ClientConstants.
                                          getAUTHOR_NAME() ));
            outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
                getActiveGroup().getWorkGroupID() );
650         try
            {
652            Data data = new Data( outgoingPackage );
               // System.out.println( "Broadcasted" );
654            groupChannel.sendToOthers( getGroupManagementClient(), data );
            }
656         catch ( JSDTException JSDTE )
            {
658            handleJSDTException( JSDTE );
            }
660      }
            if ( numClientNames > 1 )
662      {
            owner.resetMessageArea();
664      }
      }
666 }
    /**
668  * Send a difference array to the other members of the group that the client belongs to, if any, so that
        they can
     * update their copy of the relevant document. This method is called when a document has been modified
        to keep the
670  * remote copies in synch with the master at the author client.
     * Creation date: (21-08-00 16:18:43)
672  * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
     */
674 public void broadcastDocumentUpdate( String documentName, CompressedDocumentPackage
        compressedDifferenceBitmapPackage)
    {
676    DataPackage outgoingPackage = new DataPackage();

678    outgoingPackage.setContents( compressedDifferenceBitmapPackage );
```

```
      outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
          getDOCUMENT_UPDATE() );
680   outgoingPackage.getProperties().setProperty( DataPackage.getDOCPATH(), documentName );
      try
682   {
         if ( getClientInfo().getActiveGroup() != null )
684      {
            URLString url = getClientInfo().getActiveGroup().getSessionURL();
686         Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
            Channel groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
                getActiveGroup().getWorkGroupChannelID(), true, true, false );
688         groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));
         }
690   }
      catch ( JSDTException JSDTE )
692   {
         handleJSDTException( JSDTE );
694   }

696 }
    /**
698  * Set all properties of the topic tree to null so that it appears empty to the user.
     * Creation date: (01-08-00 15:53:33)
700  */
    public void clearTopicTree()
702 {
      owner.setMessage( null );
704   getClientInfo().setCurrentMessage( null );
      getClientInfo().setCurrentDocumentName( null );
706   owner.setTopicTree( null );
    }
708 /**
     * Compute a bitmap combination of two input bitmaps.
710  * Starts by XORing together the first n bytes of the two bitmaps, where n is the length of the shorter
         array.
     * Then appends the remainder of newRawFileContents, if any, to the resulting bitmap and returns it.
712  * This method is meant for use in the document update phase, both in the transmitting and receiving end
         .
     * Specifically, the sender computes the difference between an old and a new version of a document and
         broadcasts it
714  * in gzipped form to one or more recipients who then apply the below method to their old version and
         the difference
     * bitmap, ending up with a transformed document identical to the sender's version.
716  * Creation date: (21-08-00 16:18:01)
     * @return byte[]
718  * @param oldRawFileContents byte[]
     * @param newRawFileContents byte[]
720  */
    public byte[] computeBitmapCombination(byte[] oldRawFileContents, byte[] newRawFileContents)
722 {
      int shortestArrayLength;
724   int longestArrayLength;
      if ( oldRawFileContents.length < newRawFileContents.length )
726   {
         shortestArrayLength = oldRawFileContents.length;
728      longestArrayLength = newRawFileContents.length;
      }
730   else
      {
732      shortestArrayLength = newRawFileContents.length;
         longestArrayLength = oldRawFileContents.length;
734   }
      byte differenceBitmap[] = new byte[ newRawFileContents.length ];
736   // XOR the two file contents arrays and store the result in the differenceBitmap array
      int i;
738   for ( i = 0 ; i < shortestArrayLength; i++ )
```

```
      {
         byte oldByte = oldRawFileContents[i];
         byte newByte = newRawFileContents[i];
         // ( a AND b )OR ( !a AND !b )== XOR( a,b )
         differenceBitmap[i] = (byte) (
            ( (byte) ( oldByte | newByte ))&
            ( (byte) ( (Byte.MAX_VALUE - oldByte) | (Byte.MAX_VALUE - newByte ))));
      }
      // copy remainder of new file contents, if any, to differenceBitmap array
      for ( ; i < newRawFileContents.length; i++ )
      {
         differenceBitmap[i] = newRawFileContents[i];
      }
      return differenceBitmap;
   }
   /**
    * Perform various initialization maneuvers at start-up
    * Creation date: (21-10-00 21:35:37)
    */
   public void configure()
   {
      connectToGroupManagement();
      requestGroupsFromServer();
      connectToGroup();
      addDocumentsFromDisk( readDocumentList() );
   }
   /**
    * Connect to the most recently joined group, if any, as specified in the preferences.
    * Creation date: (09-08-00 23:48:29)
    */
   public void connectToGroup()
   {
      if ( getClientInfo().getGroups() != null )
      {
         String activeGroupUponExitID = (String) ClientInfo.getPreferences().getProperty( ClientConstants.
              getACTIVE_GROUP_UPON_EXIT() );
         WorkGroup workGroup = (WorkGroup) getClientInfo().getGroups().get( activeGroupUponExitID );
         if ( activeGroupUponExitID.equalsIgnoreCase( ClientConstants.getDEFAULT_GROUP_ID() )== false )
         {
            connectToGroup( workGroup );
         }
         else
         {
            getClientInfo().setActiveGroup( null );
            informUserOfLocalMode();
         }
      }
   }
   /**
    * Connect to the workgroup specified by the parameter, if possible.
    * Creation date: (10-08-00 00:35:15)
    * @param groupURL com.sun.media.jsdt.URLString
    */
   public void connectToGroup( WorkGroup workGroup )
   {
      Session session;
      Channel channel;
      try
      {
         URLString url = workGroup.getSessionURL();
         session = SessionFactory.createSession( getGroupManagementClient(), url, true );
         channel = session.createChannel( getGroupManagementClient(),
                           workGroup.getWorkGroupChannelID() ,
                           true, true, true);
         channel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
         getClientInfo().setActiveGroup( workGroup );
```

```
804        getClientInfo().setLocalMode( false );
           // broadcast to other group members
806        broadcastDocuments( getClientInfo().getLocalFiles() );
           // request documents from other members of group
808        requestDocuments();
           // announce successful join
810        informUserOfGroupJoin( workGroup );
       }
812    catch ( JSDTException JSDTE )
       {
814        handleJSDTException( JSDTE );
       }
816

818 }
    /**
820  * Connect the client to the server via the group management session.
     * Creation date: (02-07-00 19:16:42)
822  */
    public void connectToGroupManagement()
824 {
       Session session = null;
826    Channel groupManagementChannel = null;
       if ( getClientInfo().getGroupManagementSession() == null )
828    {
          try
830       {  // disconnect first if already connected
       /*    if ( getClientInfo().getGroupManagementSession() != null )
832           {
                if ( getClientInfo().isLocalMode() == false )
834             {
                   disconnectFromGroup( getClientInfo().getActiveGroup() );
836             }
                clientInfo.getGroupManagementSession().leave( getGroupManagementClient() );
838             //clientInfo.getGroupManagementChannel().leave( getGroupManagementClient() );
                clientInfo.setGroupManagementSession( null );
840             clientInfo.setGroupManagementChannel( null );
           }
842    */
           URLString url = URLString.createSessionURL( (String)ClientInfo.getPreferences().getProperty(
                ClientConstants.getSERVER_NAME() ),
844                                   Integer.parseInt( (String) (ClientInfo.getPreferences().
                                        getProperty( ClientConstants.getSERVER_PORT() ))),
                                   (String)ClientInfo.getPreferences().getProperty( ClientConstants.
                                        getCONNECTION_TYPE() ),
846                                ClientConstants.getGROUP_MANAGEMENT_SESSION_NAME() );
           if ( SessionFactory.sessionExists( url ))
848        {
              session = SessionFactory.createSession( getGroupManagementClient(), url, true);
850           groupManagementChannel = session.createChannel( getGroupManagementClient(), ClientConstants.
                   getGROUP_CHANNEL_NAME(), true, true, true);
              groupManagementChannel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
852           owner.initMessageArea( 1, 1, 1, ClientConstants.getCONNECTED_TO_GROUP_MANAGEMENT() );

854        }
           else
856        {
              informUserOfLocalMode();
858           owner.initMessageArea( 1, 1, 1, ClientConstants.getFAILED_TO_CONNECT_TO_SERVER() );
           }
860       }
          catch ( NameInUseException NIUE )
862       {
              owner.initMessageArea( 1, 1, 1, ClientConstants.getNOTIFYING_SERVER_OF_NAME_IN_USE() );
864           informServerOfExit();
          }
```

```
866         catch ( JSDTException JSDTE )
            {
868            handleJSDTException( JSDTE );
               informUserOfLocalMode();
870         }

872         getClientInfo().setGroupManagementSession(session);
            getClientInfo().setGroupManagementChannel(groupManagementChannel);
874      }
      }
876
      /**
878    * Copy the text selected in the visual component associated with the selectedNode parameter to the
           client
       * clipboard.
880    * Creation date: (07-08-00 17:01:17)
       * @param selectedNode edu.umd.cs.jazz.ZVisualLeaf
882    */
      public void copyText(edu.umd.cs.jazz.ZVisualLeaf selectedNode)
884   {
         JScrollPane selectedPane = (JScrollPane) ( (PZSwing) selectedNode.getVisualComponent() ).getComponent
             ();
886      // if the selected node is capable of containing text. The check is mostly to be on the safe side; it
             is assumed that
         // the caller tries to ensure that the copy text menu item is invoked only on text documents.
888      if ( selectedPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
         {
890         String selectedText = ( (javax.swing.text.JTextComponent) selectedPane.getViewport().getView() ).
                getSelectedText();
            if ( selectedText != null )
892         {
               ( (javax.swing.text.JTextComponent) selectedPane.getViewport().getView() ).copy();
894            owner.initMessageArea( 1, 1, 1, ClientConstants.getCOPIED_TEXT_MESSAGE() );
            }
896         else
            {
898            owner.initMessageArea( 1, 1, 1, ClientConstants.getNO_TEXT_TO_COPY_MESSAGE() );
            }
900      }
      }
902   /**
       * Submit a request to the server that a group be created and added to the group directory.
904    * Creation date: (17-07-00 17:25:50)
       * @param workGroup peerviewmisc.WorkGroup
906    */
      public void createWorkGroup(WorkGroup workGroup)
908   {
         DataPackage outgoingPackage = new DataPackage();
910
         outgoingPackage.setContents( workGroup );
912      outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getREQUEST_ADD_GROUP() );
         try
914      {
            if ( getClientInfo().getGroupManagementChannel() != null )
916         {
            getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
918                                   ClientInfo.getPreferences().getProperty( ClientConstants.
                                          getSERVER_NAME() ),
                                      new Data( outgoingPackage ));
920         }
         }
922      catch ( JSDTException JSDTE )
         {
924         handleJSDTException( JSDTE );
         }
```

```
926  }
     /**
928   * Compress and package the contents of an array representing a document.
      * Creation date: (08-07-00 20:49:16)
930   * @return CompressedDocumentPackage
      * @param uncompressedContents byte
932   */
     public CompressedDocumentPackage deflateDocument(byte[] uncompressedContents)
934  {
        Deflater deflater = new Deflater( Integer.parseInt( ClientInfo.getPreferences().getProperty(
             ClientConstants.getCOMPRESSION_LEVEL() )));
936     byte compressedContents[] = new byte[ uncompressedContents.length ];

938     deflater.setInput( uncompressedContents, 0 , uncompressedContents.length );
        deflater.finish();
940     int compressedSize = deflater.deflate( compressedContents );
        byte compressedAndCondensedContents[] = new byte[ compressedContents.length ];
942     for ( int i = 0 ; i < compressedSize; i++ )
        {
944        compressedAndCondensedContents[ i ] = compressedContents[ i ];
        }
946     CompressedDocumentPackage compressedDocumentPackage = new CompressedDocumentPackage();

948     compressedDocumentPackage.setCompressedContents( compressedAndCondensedContents );
        compressedDocumentPackage.setCompressedLength( compressedSize );
950     compressedDocumentPackage.setUncompressedLength( uncompressedContents.length );

952     // System.out.println( "Original: " + uncompressedContents.length + " Deflated: " + compressedSize );
        return compressedDocumentPackage;
954  }
     /**
956   * Leave a work group (passed as argument).
      * Creation date: (10-08-00 21:14:06)
958   * @param workGroup peerviewmisc.WorkGroup
      */
960  public void disconnectFromGroup(WorkGroup workGroup)
     {
962     if ( workGroup != null )
        {
964     // broadcastDocumentRemovals( getClientInfo().getLocalFiles() );
           // the next step is carried out to allow the broadcasts to conclude - they are apparently
                disrupted if the leave is executed prematurely
966        removeDistributedDocumentsFromPanorama();
           // obtain reference to the session that this client currently subscribes to and leave it. This
                will automatically
968        // expel the client from any objects subordinate to that session.
           try
970        {
              URLString url = workGroup.getSessionURL();
972           Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
              session.leave( getGroupManagementClient() );
974        }
           catch ( JSDTException JSDTE )
976        {
              System.err.println( JSDTE.toString() );
978           System.exit( 0 );
           }
980     }
     }
982  /**
      * Insert the method's description here.
984   * Creation date: (02-07-00 19:07:18)
      * @return clientapp.ClientInfo
986   */
     public ClientInfo getClientInfo() {
988     if ( clientInfo == null )
```

```
      {
990      clientInfo = new ClientInfo();
      }
992   return clientInfo;
   }
994 /**
    * Insert the method's description here.
996 * Creation date: (02-07-00 22:53:11)
    * @return clientapp.GroupManagementClient
998 */
   public GroupManagementClient getGroupManagementClient() {
1000  if (groupManagementClient == null)
      {
1002     groupManagementClient = new GroupManagementClient();
         groupManagementClient.setName( ClientInfo.getPreferences().getProperty( ClientConstants.getNAME()
               ));
1004  }
      return groupManagementClient;
1006 }
   /**
1008 * Insert the method's description here.
    * Creation date: (09-10-00 12:53:32)
1010 * @param ce com.sun.media.jsdt.event.ConnectionEvent
    */
1012 public void groupManagementConnectionFailed(ConnectionEvent ce)
   {
1014  owner.initMessageArea( 1, 1, 1, ClientConstants.getCONNECTION_FAILURE( ce.toString() ));

1016
   }
1018 /**
    * Handle the different subclasses of JSDTException, i.e. the different types of exception that JSDT
        actions may throw,
1020 * by issuing appropriate messages and taking corrective or conclusive action.
    * Creation date: (08-08-00 16:34:28)
1022 * @param exception com.sun.media.jsdt.JSDTException
    */
1024 public void handleJSDTException(JSDTException exception)
   {
1026  String message = (String) ClientConstants.getJSDT_EXCEPTION_MESSAGES_TABLE().get( exception.getClass
         ().toString() );
      message += "␣(␣" + exception.toString() + "␣)";
1028  System.err.println( message );
      owner.initMessageArea( 1, 1, 1, message );
1030 }
   /**
1032 * Inflate the contents of a compressed document.
    * Creation date: (08-07-00 01:56:27)
1034 */
   public byte[] inflateDocument( CompressedDocumentPackage compressedDocumentPackage )
1036 {
      byte uncompressedData[] = new byte[ compressedDocumentPackage.getUncompressedLength() ];
1038  try
      {
1040     Inflater inflater = new Inflater();
         inflater.setInput( compressedDocumentPackage.getCompressedContents() );
1042     int inflatedSize = inflater.inflate( uncompressedData );
      }
1044  catch ( DataFormatException DFE )
      {
1046     System.err.println( DFE.toString() );
         System.exit( 0 );
1048  }
      // System.out.println( "Deflated: " + compressedDocumentPackage.getCompressedLength() + " Inflated
           : " + compressedDocumentPackage.getUncompressedLength() );
1050  return uncompressedData;
```

```
}
/**
 * Inform other members of the group that the client belongs to that the document designated by the
     parameter should
 be removed from their panoramas.
 * Creation date: (15-07-00 15:46:56)
 * @param fullyQualifiedDocumentName java.lang.String
 */
public void informGroupMembersOfDocumentRemoval(String fullyQualifiedDocumentName)
{
   DataPackage outgoingPackage = new DataPackage();

   outgoingPackage.setContents( fullyQualifiedDocumentName );
   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
       getCLIENT_REMOVED_LOCAL_DOCUMENTS() );
   try
   {
      if ( getClientInfo().isLocalMode() == false )
      {
         URLString url = clientInfo.getActiveGroup().getSessionURL();
         Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
         Channel channel = session.createChannel( getGroupManagementClient(), clientInfo.getActiveGroup()
             .getWorkGroupChannelID(), true, true, false );
         channel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));
      }
   }
   catch ( JSDTException JSDTE )
   {
      handleJSDTException( JSDTE );
   }
}
/**
 * Inform other group members that the panorama has changed and must be updated.
 * Creation date: (31-08-00 19:11:21)
 */
public void informGroupMembersOfPanoramaUpdate()
{
   if ( getClientInfo().isLocalMode() == false )
   {
      URLString url;
      Session session;
      Channel groupChannel = null;
         try
      {
         url = getClientInfo().getActiveGroup().getSessionURL();
         session = SessionFactory.createSession( getGroupManagementClient(), url, false );
            groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
                getActiveGroup().getWorkGroupChannelID(), true, true, false);

         DataPackage outgoingPackage = new DataPackage();
         outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getUPDATE_PANORAMA() );
         groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));

      }
      catch ( JSDTException JSDTE )
      {
         handleJSDTException( JSDTE );
      }
   }
}
/**
 * Notify the server that this client has terminated.
 * Creation date: (01-11-00 01:16:44)
 */
public void informServerOfExit()
```

```
1112  {
        DataPackage outgoingPackage = new DataPackage();
1114
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getCLIENT_LEFT() );
1116    try
        {
1118      if ( getClientInfo().getGroupManagementChannel() != null )
          {
1120        getClientInfo().getGroupManagementChannel().sendToClient(
              getGroupManagementClient(),
1122          ClientInfo.getPreferences().getProperty( ClientConstants.getSERVER_NAME() ),
              new Data( outgoingPackage ));
1124      }
        }
1126    catch ( JSDTException JSDTE )
        {
1128      handleJSDTException( JSDTE );
        }
1130  }
      /**
1132   * Inform the server that the client has exited the group to which it belonged, if any.
       * Creation date: (31-10-00 23:55:01)
1134   */
      public void informServerOfGroupExit()
1136  {
        if ( getClientInfo().getActiveGroup() != null )
1138    {
          DataPackage outgoingPackage = new DataPackage();
1140
          outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
              getCLIENT_LEFT_GROUP() );
1142      outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
              getActiveGroup().getWorkGroupID() );
          try
1144      {
            if ( getClientInfo().getGroupManagementChannel() != null )
1146        {
              getClientInfo().getGroupManagementChannel().sendToClient(
1148            getGroupManagementClient(),
                ClientInfo.getPreferences().getProperty( ClientConstants.getSERVER_NAME() ),
1150            new Data( outgoingPackage ));
            }
1152      }
          catch ( JSDTException JSDTE )
1154      {
            handleJSDTException( JSDTE );
1156      }
        }
1158  }
      /**
1160   * Inform the server that the client has added one or more documents to the panorama.
       * Creation date: (04-10-00 11:39:50)
1162   * @param docPath java.lang.String
       */
1164  public void informServerOfNewDocument(String docPath)
      {
1166    DataPackage outgoingPackage = new DataPackage();

1168    outgoingPackage.setContents( docPath );
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getCLIENT_ADDED_DOCUMENTS() );
1170    outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ClientInfo.getPreferences
            ().getProperty( ClientConstants.getAUTHOR_NAME() ));
        try
1172    {
```

```
         if ( getClientInfo().getGroupManagementChannel() != null )
1174     {
            getClientInfo().getGroupManagementChannel().sendToClient(
1176           getGroupManagementClient(),
               ClientInfo.getPreferences().getProperty( ClientConstants.getSERVER_NAME() ),
1178           new Data( outgoingPackage ));
         }
1180  }
      catch ( JSDTException JSDTE )
1182  {
         handleJSDTException( JSDTE );
1184  }
   }
1186 /**
      * Display a message in the client progress bar indicating the client has joined a specific work group.
1188  * Creation date: (26-07-00 12:34:45)
      * @param workGroup peerviewmisc.WorkGroup
1190  */
   public void informUserOfGroupJoin(WorkGroup workGroup)
1192 {
      // if this client is the only group member
1194  if ( workGroup.getParticipants().size() <= 1 )
      {
1196     owner.initMessageArea( 1, 1, 1, ClientConstants.getCLIENT_JOINED_GROUP_MESSAGE( workGroup ));
      }
1198  else
      {
1200     owner.initMessageArea( 1, 1, 1, ClientConstants.getCLIENT_JOINED_GROUP_WITH_MEMBERS_MESSAGE(
            workGroup ));
      }
1202 }
   /**
1204  * Display a notification message that the client is operating in local mode, i.e disconnected from the
         server.
      * Creation date: (10-08-00 04:34:38)
1206  */
   public void informUserOfLocalMode()
1208 {
      if ( getClientInfo().isLocalMode() == false )
1210  {
         MessageBox messageBox = new MessageBox();
1212
         messageBox.setText( ClientConstants.getLOCAL_MODE_MESSAGE() );
1214     messageBox.show();
         getClientInfo().setLocalMode( true );
1216  }
   }
1218 /**
      * Insert the method's description here.
1220  * Creation date: (04-07-00 17:02:27)
      */
1222 public void initialize()
   {
1224    clearTopicTree();
   }
1226 /**
      * Leave the group management communication channel.
1228  * Creation date: (20-10-00 15:43:39)
      */
1230 public void leaveGroupManagement()
   {
1232    try
      {
1234     if ( getClientInfo().getGroupManagementChannel() != null )
         {
1236        getClientInfo().getGroupManagementChannel().leave( getGroupManagementClient() );
```

```
         }
1238     if ( getClientInfo().getGroupManagementSession() != null )
         {
1240        getClientInfo().getGroupManagementSession().leave( getGroupManagementClient() );
         }
1242  }
      catch ( JSDTException JSDTE )
1244  {
         handleJSDTException( JSDTE );
1246  }
   }
1248 /**
      * Perform termination housekeeping such as updating preferences and
1250  * sundry other chores.
      * Creation date: (09-08-00 23:23:16)
1252  */
   public void prepareForTermination()
1254 {
      java.awt.Dimension size = owner.getSize();
1256  java.awt.Point location = owner.getLocation();

1258  ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_HORIZONTAL_SIZE(),
                              String.valueOf( (int)size.getWidth() ));
1260  ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_VERTICAL_SIZE(),
                              String.valueOf( (int)size.getHeight() ));
1262  ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_X_COORDINATE(),
                              String.valueOf( (int)location.getX() ));
1264  ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_Y_COORDINATE(),
                              String.valueOf( (int)location.getY() ));
1266  if ( getClientInfo().isLocalMode() == false )
      {
1268     ClientInfo.getPreferences().setProperty( ClientConstants.getACTIVE_GROUP_UPON_EXIT(),
                              getClientInfo().getActiveGroup().getWorkGroupID() );
1270  }
      getClientInfo().writePreferences();
1272  writeDocumentList();
      leaveGroupManagement();
1274  owner.getPanoramaPanel().getUpdateTimer().stop();
      try
1276  {
         System.runFinalization();
1278     Runtime.getRuntime().exit(0);
      }
1280  catch ( Exception E )
      {
1282     System.out.println( ClientConstants.getCOULD_NOT_EXIT_PROPERLY() );
      }
1284 }
   /**
1286 * Creation date: (14-06-00 22:34:24)
      * @param newFilenames java.lang.File[]
1288  */
   public byte[] readDocument(File newFile) throws IOException
1290 {
      DataInputStream input;
1292  byte rawFileContents[];
      // fetch contents of file as raw array of bytes
1294  rawFileContents = new byte[ (int)newFile.length() ];
      try
1296  {  input = new DataInputStream( new FileInputStream(newFile) );
         input.read( rawFileContents );
1298  }
      catch (IOException e)
1300  {  System.err.println( ClientConstants.failedToAccessDocument( newFile.toString() ));
         System.exit(1); //SKALÆ NDRES: fejlmeddelelse øbr fremkomme i status area i client window
1302  }
```

```
      return rawFileContents;
1304  }
      /**
1306   * Read the list of documents from disk and insert the file names it contains into a vector.
       * Creation date: (06-09-00 21:17:15)
1308   */
      public java.util.Vector readDocumentList()
1310  {
      java.util.Vector docs = null;
1312  try
      {
1314  FileInputStream fis = new FileInputStream( ClientConstants.getDOCUMENT_LIST_FILENAME() );
      ObjectInputStream ois = new ObjectInputStream( fis );
1316  try
      {
1318     docs = (java.util.Vector) ois.readObject();
      }
1320  catch ( ClassNotFoundException CNFE )
      {
1322     System.err.println( CNFE.toString() );
      }
1324  ois.close();
      }
1326  catch ( IOException ioe )
      {
1328     System.err.println( ClientConstants.getCOULD_NOT_READ_DOCUMENT_LIST() );
         owner.initMessageArea( 1, 1, 1, ClientConstants.getCOULD_NOT_READ_DOCUMENT_LIST() );
1330  }
      return docs;
1332  }
      /**
1334   * Remove from the panorama all documents originating from other members of the group to which this
          client currently belongs.

1336   * Creation date: (15-07-00 16:23:13)
       * @param senderName java.lang.String
1338   * @param fullyQualifiedDocumentName java.lang.String
       */
1340  public void removeDistributedDocument(String authorName, String senderName, String documentPath)
      {
1342  owner.initMessageArea( 1, 1, 1, ClientConstants.getREMOVING_DISTRIBUTED_DOCUMENT( documentPath,
          authorName ));
      owner.getPanoramaPanel().removeGlobalDocument( senderName, documentPath );
1344  owner.getPanoramaPanel().revalidate();
      owner.getPanoramaPanel().repaint();
1346  }
      /**
1348   * Remove all distributed documents from the local panorama and clear the hashtable indexing them.
       * Creation date: (26-07-00 02:54:40)
1350   */
      public void removeDistributedDocumentsFromPanorama()
1352  {
      ( new RemoveDistributedDocumentsFromPanorama() ).start();
1354  owner.getPanoramaPanel().zoomToOverview();
      }
1356  /**
       * Launch a removal process for a set of documents in a separate thread (to allow concurrent updating of
           the GUI).
1358   * Creation date: (03-10-00 11:23:07)
       * @param removables java.io.File[]
1360   */
      public void removeDocuments(File[] removables)
1362  {
      ( new RemoveDocuments( removables )).start();
1364  }
      /**
```

```
1366  * Submit a request to the server that it removes a specific group from the group directory.
      * Creation date: (26-07-00 18:57:18)
1368  * @param workGroup peerviewmisc.WorkGroup
      */
1370 public void removeWorkGroup(WorkGroup workGroup)
     {
1372    DataPackage outgoingPackage = new DataPackage();

1374    outgoingPackage.setContents( workGroup );
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getREQUEST_REMOVE_GROUP() );
1376    try
        {
1378       if ( getClientInfo().getGroupManagementChannel() != null )
           {
1380       getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                                          ClientInfo.getPreferences().getProperty( ClientConstants.
                                              getSERVER_NAME() ),
1382                                      new Data( outgoingPackage ));
           }
1384    }
        catch ( JSDTException JSDTE )
1386    {
           handleJSDTException( JSDTE );
1388    }

1390 }
     /**
1392  * Request copy of documents from all other members of active group.
      * Creation date: (08-07-00 20:13:19)
1394  */
     public void requestDocuments()
1396 {
        if ( getClientInfo().isLocalMode() == false )
1398    {
           WorkGroup activeGroup = getClientInfo().getActiveGroup();
1400       if ( activeGroup.getDocuments().size() > 0 )
           {
1402          // System.out.println( "Requested documents" );
              //owner.initMessageArea(  1, activeGroup.getDocuments().size() - getClientInfo().getLocalFiles()
                  .size(), 1,
1404          //              ClientConstants.getREQUESTING_DOCUMENTS_MESSAGE( activeGroup.getDocuments().
                  size() - getClientInfo().getLocalFiles().size() ));

1406          DataPackage outgoingPackage = new DataPackage();
              outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
                  getREQUEST_DOCUMENTS() );
1408          outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ClientInfo.
                  getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ));
              try
1410          {
                 URLString url = activeGroup.getSessionURL();
1412             Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
                 Channel groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
                     getWorkGroupChannelID(), true, true, false );
1414
                 groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));
1416          }
              catch ( JSDTException JSDTE )
1418          {
              handleJSDTException( JSDTE );
1420          }
           }
1422    }
     }
1424 /**
```

```
     * Request the directory of groups from the server. Assumes that group management is up and running.
1426  * The directory will be received by the groupManagementClient data member.
     * Creation date: (04-07-00 16:27:33)
1428  */
    public void requestGroupsFromServer()
1430 {
       DataPackage outgoingPackage = new DataPackage();
1432   outgoingPackage.setContents( null );
       outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getREQUEST_GROUP_DIRECTORY() );
1434   try
       {
1436      if ( getClientInfo().getGroupManagementChannel() != null )
          {
1438         for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i
                 ++)
             {
1440            // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
             }
1442      getClientInfo().getGroupManagementChannel().sendToClient( groupManagementClient,
                                                ClientInfo.getPreferences().getProperty( ClientConstants.
                                                     getSERVER_NAME() ),
1444                                            new Data( outgoingPackage ));
          }
1446   }
       catch ( JSDTException JSDTE )
1448   {
          handleJSDTException( JSDTE );
1450   }
    }
1452 /**
     * Request a specific message from the message database maintained by the server.
1454  * Creation date: (10-07-00 02:11:23)
     * @param messageID java.lang.String
1456  */
    public void requestMessageFromServer(String messageID)
1458 {
       DataPackage outgoingPackage = new DataPackage();
1460
       outgoingPackage.setContents( null );
1462   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getREQUEST_MESSAGE() );
       outgoingPackage.getProperties().setProperty( DataPackage.getMESSAGE_ID(), messageID );
1464   try
       {
1466      if ( getClientInfo().getGroupManagementChannel() != null )
          {
1468         for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i
                 ++)
             {
1470            // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
             }
1472         clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                                                 ClientInfo.getPreferences().getProperty( ClientConstants.
                                                      getSERVER_NAME() ),
1474                                             new Data( outgoingPackage ));
          }
1476   }
       catch ( JSDTException JSDTE )
1478   {
          handleJSDTException( JSDTE );
1480   }
    }
1482 /**
     * Request that the server submit the current topic tree for the document designated by documentName.
1484  * Creation date: (09-07-00 18:35:18)
```

```
     * @param documentName java.lang.String
1486  */
     public void requestTopicTree(String documentName)
1488 {
        System.out.println( "Inde␣i␣requestTopicTree" );
1490    DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents( documentName );
1492    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getREQUEST_TOPIC_TREE() );
        try
1494    {
           if ( getClientInfo().getGroupManagementChannel() != null )
1496       {
           getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1498                                          ClientInfo.getPreferences().getProperty( ClientConstants.
                                                   getSERVER_NAME() ),
                                              new Data( outgoingPackage ));
1500       }
        }
1502    catch ( JSDTException JSDTE )
        {
1504       handleJSDTException( JSDTE );
        }
1506 }
     /**
1508  * Notify the server that the client added a message.
      * Creation date: (11-07-00 21:23:22)
1510  * @param newMessage peerviewmisc.Message
      */
1512 public void sendMessage(Message newMessage)
     {
1514    DataPackage outgoingPackage = new DataPackage();

1516    outgoingPackage.setContents( newMessage );
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getCLIENT_ADDED_MESSAGE() );
1518    try
        {
1520       if ( getClientInfo().getGroupManagementChannel() != null )
           {
1522          clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                                                 ClientInfo.getPreferences().getProperty( ClientConstants.
                                                      getSERVER_NAME() ),
1524                                             new Data( outgoingPackage ));
           }
1526    }
        catch ( JSDTException JSDTE )
1528    {
           handleJSDTException( JSDTE );
1530    }
     }
1532 /**
      * Submit a request to the server that a specific message be deleted.
1534  * Creation date: (12-07-00 12:05:06)
      * @param messageID java.lang.String
1536  */
     public void sendMessageDeletion(String messageID)
1538 {
        DataPackage outgoingPackage = new DataPackage();
1540
        outgoingPackage.setContents( messageID );
1542    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getDELETE_MESSAGE() );
        try
1544    {
           if ( getClientInfo().getGroupManagementChannel() != null )
```

```
1546        {
               clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1548                                                ClientInfo.getPreferences().getProperty( ClientConstants.
                                                        getSERVER_NAME() ),
                                                    new Data( outgoingPackage ));
1550        }
         }
1552     catch ( JSDTException JSDTE )
         {
1554        handleJSDTException( JSDTE );
         }
1556  }
      /**
1558   * Sibmit an updated topic tree to the server for inclusion in its topic tree database.
       * Creation date: (11-07-00 21:23:37)
1560   */
      public void sendTopicTreeUpdate( DefaultTreeModel newTree )
1562  {
         DataPackage outgoingPackage = new DataPackage();
1564
         outgoingPackage.setContents( newTree );
1566     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
             getTOPIC_TREE_UPDATE() );
         outgoingPackage.getProperties().setProperty( DataPackage.getDOCPATH(), getClientInfo().
             getCurrentDocumentName() );
1568     try
         {
1570        if ( getClientInfo().getGroupManagementChannel() != null )
            {
1572        clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
                                                ClientInfo.getPreferences().getProperty( ClientConstants.
                                                        getSERVER_NAME() ),
1574                                            new Data( outgoingPackage ));
            }
1576     }
         catch ( JSDTException JSDTE )
1578     {
            handleJSDTException( JSDTE );
1580     }
      }
1582  /**
       * Insert the method's description here.
1584   * Creation date: (02-07-00 19:07:18)
       * @param newClientInfo clientapp.ClientInfo
1586   */
      public void setClientInfo(ClientInfo newClientInfo) {
1588     clientInfo = newClientInfo;
      }
1590  /**
       * Update the text area for message display in the discussion panel at the bottom of the client main
           window.
1592   * Creation date: (10-07-00 02:30:47)
       * @param newMessage peerviewmisc.Message
1594   */
      public void setDisplayedMessage(Message newMessage)
1596  {
         owner.setMessage( newMessage );
1598     getClientInfo().setCurrentMessage( newMessage );
      }
1600  /**
       * Insert the method's description here.
1602   * Creation date: (02-07-00 22:53:11)
       * @param newGroupManagementClient clientapp.GroupManagementClient
1604   */
      public void setGroupManagementClient(GroupManagementClient newGroupManagementClient) {
1606     groupManagementClient = newGroupManagementClient;
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (09-07-00 22:48:06)
 * @param newTree javax.swing.tree.DefaultTreeModel
 * @param documentName java.lang.String
 */
public void setTopicTree(DefaultTreeModel newTree, String documentName)
{
   setDisplayedMessage( null );
   getClientInfo().setCurrentDocumentName( documentName );
   owner.setTopicTree( newTree );
}
/**
 * Update one of the distributed documents shown in the panorama.
 * Creation date: (21-08-00 17:30:56)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
 */
public void updateDistributedDocument(String documentName, String senderName, CompressedDocumentPackage
     compressedDifferenceBitmapPackage)
{
   owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATING_DISTRIBUTED_DOCUMENT( documentName ));
   byte rawFileContents[] = inflateDocument( compressedDifferenceBitmapPackage );
   owner.getPanoramaPanel().updateDistributedDocument( senderName, documentName, rawFileContents );
}
/**
 * Extract difference bitmap by XORing old and new file contents. Then submit the result as a deflated,
      i.e. gzipped,
 * data package to other members, if any, of active group.
 * Creation date: (21-08-00 16:01:13)
 * @param path java.lang.String
 * @param clientName java.lang.String
 * @param oldRawFileContents byte[]
 * @param newRawFileContents byte[]
 */
public void updateDocument(String path, String clientName, byte[] oldRawFileContents, byte[]
     newRawFileContents)
{
   byte differenceBitmap[] = computeBitmapCombination( oldRawFileContents, newRawFileContents );
   CompressedDocumentPackage compressedDifferenceBitmapPackage = deflateDocument( differenceBitmap );
   broadcastDocumentUpdate( path, compressedDifferenceBitmapPackage );
}
/**
 * Insert the method's description here.
 * Creation date: (04-08-00 15:50:41)
 * @param newDirectory java.util.Hashtable
 */
public void updateGroupDirectory(Hashtable newDirectory)
{
   getClientInfo().setGroups( newDirectory );
   owner.getGroupDirectory1().updateTable();
   if ( getClientInfo().getActiveGroup() != null )
   {
      getClientInfo().setActiveGroup( (WorkGroup) newDirectory.get( getClientInfo().getActiveGroup().
           getWorkGroupID() ));
   }
}
/**
 * Insert the method's description here.
 * Creation date: (14-06-00 23:07:20)
 */
public void updatePanorama()
{
   owner.updateVisibleDocumentsBox();
```

```
      // System.out.println("updatePanorama");
      owner.getPanoramaPanel().validateLayout();
   }
   /**
    * Insert the method's description here.
    * Creation date: (31-08-00 02:14:49)
    */
   public void updatePanoramaByRequest()
   {
      owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() );
      updatePanorama();
   }
   /**
    * Insert the method's description here.
    * Creation date: (27-08-00 21:21:06)
    * @param fullyQualifiedDocumentName java.lang.String
    * @param treeUpdate javax.swing.tree.DefaultTreeModel
    */
   public void updateTopicTree(String senderName, String documentName, DefaultTreeModel treeUpdate)
   {
      String fqName = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
      if ( getClientInfo().getCurrentDocumentName().equals( fqName ))
      {
         setTopicTree( treeUpdate, fqName );
      }
   }
   /**
    * Test connection to group management and notify user if it isn't active.
    * Creation date: (09-10-00 18:57:46)
    * @return boolean
    */
   public boolean verifyGroupConnectionManagement()
   {
      if ( getClientInfo().getGroupManagementSession() == null )
      {
         owner.initMessageArea( 1, 1, 1, ClientConstants.getNOT_CONNECTED_TO_SERVER() );
         return false;
      }
      else
      {
         return true;
      }
   }
   /**
    * Extract a list of documents displayed in the panorama and write it to disk. This can then be restored
        automatically
    * the next time the client is started.
    * Creation date: (06-09-00 21:48:43)
    */
   public void writeDocumentList()
   {
      try
      {
         FileOutputStream fos = new FileOutputStream( ClientConstants.getDOCUMENT_LIST_FILENAME() );
         ObjectOutputStream oos = new ObjectOutputStream( fos );

         java.util.Vector docs = new java.util.Vector();
         Iterator iterator = owner.getPanoramaPanel().getLocalDocuments().values().iterator();
         while ( iterator.hasNext() )
         {
            edu.umd.cs.jazz.ZVisualLeaf leaf = (edu.umd.cs.jazz.ZVisualLeaf) iterator.next();
            DocPackage docPackage = new DocPackage( (String) leaf.getClientProperty( ClientConstants.
                 getDOCUMENT_NAME() ),
                                       new java.awt.Dimension( (int) leaf.getVisualComponent().getBounds().
                                            getMaxX(),
```

```
                                                    (int) leaf.getVisualComponent().getBounds().getMaxY
                                                        () ),
                                    new java.awt.Dimension( (int) leaf.editor().getTransformGroup().
                                        getTranslateX(),
                                                    (int) leaf.editor().getTransformGroup().
                                                        getTranslateY() )
                                    );
            docPackage.setNodeLocked( (String) leaf.getClientProperty( ClientConstants.getNODE_LOCKED() ));
            docs.add( docPackage );
         }
         oos.writeObject( docs );
         oos.flush();
         oos.close();
      }
      catch ( IOException ioe )
      {
         System.err.println( ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
         owner.initMessageArea( 1, 1, 1, ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
      }

   }
   }
```

Listing C.6: ClientConstants.java

```
package clientapp;

 import java.awt.Color;
 import java.util.Hashtable;
 import edu.umd.cs.jazz.*;
 import edu.umd.cs.jazz.component.*;
 import edu.umd.cs.jazz.event.*;
 import edu.umd.cs.jazz.io.*;
 import edu.umd.cs.jazz.util.*;
 import java.util.Properties;
 import peerviewmisc.SharedConstants;
 import java.util.Random;

 /**
 * This class contains constants specific to the client application as well as those derived from the {
     @link #peerviewmisc.SharedConstants SharedConstants} class.
 * Creation date: (15-06-00 22:57:59)
 * @author:
 */
public class ClientConstants extends SharedConstants {

   private final static Color BACKGROUND_COLOUR = new Color(200, 200, 200);
   public final static java.lang.String PROCESSING_DOCUMENTS = "Adding␣documents␣to␣the␣panorama";
   public final static java.lang.String MESSAGE_AREA_DEFAULT_STRING = "Message␣area:␣double-click␣here␣
       to␣open␣in␣separate␣window";
   private final static int UPDATE_INTERVAL_IN_MILLISECS = 1;
   private final static java.lang.String RETRIEVING_DOCUMENTS = "Adding␣documents␣to␣panorama";

   private final static java.lang.String PLAIN_TEXT = "text/plain";
   private final static java.lang.String HTML = "text/html";
   private final static java.lang.String RTF = "text/rtf";
   private final static java.lang.String BITMAP = "bitmap";
   private final static java.lang.String ALL_FILES = "All␣files";
   private final static java.lang.String[][] FileTypes = {  {".txt", "Plain␣text␣(.txt)", PLAIN_TEXT},
                                          {".html", "HTML␣(.html,␣.htm)", HTML},
                                          {".rtf", "RTF␣(.rtf)", RTF},
                                          {".cpp", "C++␣source␣(.cpp,␣.cc)", PLAIN_TEXT},
                                          {".h", "C++␣header␣(.h)", PLAIN_TEXT},
                                          {".java", "Java␣source␣(.java)", PLAIN_TEXT},
                                          {".gif", "GIF␣(.gif)", BITMAP},
                                          {".jpg", "JPG␣(.jpg)", BITMAP} };
```

```
42    private final static java.lang.String[][] FILE_TYPE_SYNONYM_TABLE = {  {".txt"},
                                                 {".html", ".htm" },
44                                               {".rtf"},
                                                 {".cpp", ".cc"},
46                                               {".h"},
                                                 {".java"},
48                                               {".gif"},
                                                 {".jpg"}
50                                             };

52    private final static java.lang.String OUT_OF_BOUNDS_EXCEPTION_TEXT = "Could␣not␣add␣all␣documents␣to␣
          panorama␣due␣to␣layout␣restrictions";
      private final static int FILE_EXTENSION_INDEX = 0;
54    private final static int FILE_TYPE_DESCRIPTION_INDEX = 1;
      private final static int TYPE_INDEX = 2;
56
      private final static java.lang.String REMOVING_DOCUMENTS = "Removing␣documents␣from␣panorama";
58
      private final static java.lang.String[][] DISPLAY_QUALITIES = { {"Low", String.valueOf(
          ZDrawingSurface.RENDER_QUALITY_LOW)},
60                                               {"Medium", String.valueOf(ZDrawingSurface.
                                                     RENDER_QUALITY_MEDIUM)},
                                                 {"High", String.valueOf(ZDrawingSurface.
                                                     RENDER_QUALITY_HIGH)} };
62
      public final static java.lang.String GRID_LAYOUT = "Grid␣layout";
64
      public final static java.lang.String UPDATE_INTERVAL = "Update␣interval";
66    private final static java.lang.String NAME = "Unique␣client␣ID";
      private final static java.lang.String SIGNATURE = "Signature";
68    private final static java.lang.String DISPLAY_QUALITY = "Display␣quality";
      private final static java.lang.String LAYOUT_SCHEME = "Layout␣scheme";
70    public final static java.lang.String LOCAL_HOST = "localhost";
      public final static java.lang.String ANIMATION_DURATION_IN_MILLISECS = "Animation␣duration";
72    private final static java.lang.String AUTHOR_NAME = "Author␣name";
      private final static java.lang.String DEFAULT_DOCUMENT_WIDTH = "Default␣document␣width";
74    private final static java.lang.String DEFAULT_DOCUMENT_HEIGHT = "Default␣document␣height";
      private final static java.lang.String DEFAULT_HORIZONTAL_DOCUMENT_SPACING = "Default␣horizontal␣
          document␣spacing";
76    private final static java.lang.String DEFAULT_VERTICAL_DOCUMENT_SPACING = "Default␣vertical␣document␣
          spacing";
      private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS = "Grid␣layout␣default␣
          number␣of␣rows";
78    private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS = "Grid␣layout␣default␣
          number␣of␣columns";
      private final static java.lang.String FRAME_HORIZONTAL_SIZE = "Frame␣horizontal␣size";
80    private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame␣vertical␣size";
      private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";
82    private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";
      private final static java.lang.String FRAME_X_COORDINATE = "Frame␣x␣coordinate";
84    private final static java.lang.String FRAME_Y_COORDINATE = "Frame␣y␣coordinate";
      private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";
86    private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";
      private final static java.lang.String ACTIVE_GROUP_UPON_EXIT = "Active␣group␣upon␣exit";
88    private final static java.lang.String DEFAULT_GROUP_ID = "";
      private final static java.lang.String ZOOM_TO_OVERVIEW_ON_UPDATE = "Zoom␣to␣overview␣on␣update";
90    private final static java.lang.String ZOOM_TO_OVERVIEW = "zoom␣to␣overview␣on␣update";
      private final static java.lang.String DO_NOT_ZOOM_TO_OVERVIEW = "do␣not␣zoom␣to␣overview";
92    private final static java.lang.String SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX = "Show␣full␣
          pathnames␣in␣visible␣documents␣box";
      public final static java.lang.String TRUE = "true";
94    public final static java.lang.String FALSE = "false";
      private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH = "Default␣grid␣layout␣
          document␣width";
```

```
96      private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT = "Default␣grid␣layout␣
            document␣height";
        private final static java.lang.String DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING = "Default␣grid␣layout␣
            horizontal␣spacing";
98      private final static java.lang.String DEFAULT_GRID_LAYOUT_VERTICAL_SPACING = "Default␣grid␣layout␣
            vertical␣spacing";
        private final static java.lang.String LOAD_DOCUMENT_LIST_ON_STARTUP = "Load␣document␣list␣on␣startup"
            ;
100     private final static java.lang.String[][] DEFAULT_PREFERENCES = { {UPDATE_INTERVAL, "5"},
                                                {NAME, DEFAULT_CLIENT_NAME},
102                                             {AUTHOR_NAME, "?"},
                                                {SIGNATURE, ""},
104                                             {DISPLAY_QUALITY, "High"},
                                                {LAYOUT_SCHEME, GRID_LAYOUT},
106                                             {SERVER_NAME, DEFAULT_SERVER_NAME},
                                                {SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT)},
108                                             {CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE},
                                                {ANIMATION_DURATION_IN_MILLISECS, "1000"},
110                                             {DEFAULT_DOCUMENT_WIDTH, "400"},
                                                {DEFAULT_DOCUMENT_HEIGHT, "500"},
112                                             {DEFAULT_HORIZONTAL_DOCUMENT_SPACING, "500"},
                                                {DEFAULT_VERTICAL_DOCUMENT_SPACING, "500"},
114                                             {GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS, "10"},
                                                {GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS, "10"},
116                                             {FRAME_HORIZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE},
                                                {FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE},
118                                             {FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE},
                                                {FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE},
120                                             {ACTIVE_GROUP_UPON_EXIT, DEFAULT_GROUP_ID},
                                                {ZOOM_TO_OVERVIEW_ON_UPDATE, ZOOM_TO_OVERVIEW},
122                                             {getCOMPRESSION_LEVEL(), String.valueOf(
                                                    getDEFAULT_COMPRESSION_LEVEL() )},
                                                {SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX, FALSE},
124                                             {DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH, "500"},
                                                {DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT, "600"},
126                                             {DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING, "100"},
                                                {DEFAULT_GRID_LAYOUT_VERTICAL_SPACING, "100"},
128                                             {LOAD_DOCUMENT_LIST_ON_STARTUP, getTRUE()},
                                                {getREGISTRY_PORT(), getDEFAULT_REGISTRY_PORT()},
130                                             {getAUTOLOCK_NODES(), getFALSE()},
                                                {getMESSAGE_LOG_FILENAME(), "message.log"},
132                                           };

134     private final static java.lang.String PREFERENCES_FILE_NAME = "prefs.ini";
        private final static java.lang.String[] LAYOUT_SCHEMES = {"Grid␣layout"};
136     private final static java.lang.String PROPERTIES_HEADER = "PeerView␣client␣settings";
        private final static double DEFAULT_PANORAMA_PERCENTUAL_SIZE = 100;
138
        private final static int LOCAL_DOCUMENT = 0;
140     public final static java.lang.String DISTRIBUTED_DOCUMENT = "Distributed␣document";
        private final static java.lang.String REQUEST_DOCUMENTS = "Request␣documents";
142     private final static java.lang.String DOCUMENT_PATH = "Document␣path";
        private final static java.lang.String NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE = "You␣must␣select␣an␣
            item␣from␣the␣topic␣tree." ;
144     private final static double SCALE_FACTOR_SCALING = 100.0;
        public final static java.lang.String INVALID_DELETION_ATTEMPTED_MESSAGE = "You␣can␣only␣delete␣
            messages␣that␣are␣composed␣by␣you␣and␣which␣have␣not␣yet␣been␣responded␣to.";
146     public final static java.lang.String MESSAGE_PROPERTIES_NAME_COLUMN_HEADING = "Name";
        public final static java.lang.String MESSAGE_PROPERTIES_VALUE_COLUMN_HEADING = "Value";
148     private final static int GROUP_DIRECTORY_COLUMN_WIDTH = 250;
        public final static java.lang.String NO_GROUP_SELECTED_TO_JOIN_MESSAGE = "You␣must␣select␣a␣group␣
            from␣the␣list␣before␣joining.";
150     private final static java.lang.String GROUP_AT_MAXIMUM_MEMBERS_MESSAGE = "You␣cannot␣join␣the␣
            selected␣group␣because␣it␣has␣reached␣its␣maximum␣number␣of␣participants";
        public final static java.lang.String ALREADY_JOINED_TO_GROUP_MESSAGE = "You␣have␣already␣joined␣the␣
            selected␣group.";
```

```
152     public final static java.lang.String BROADCASTING_DOCUMENT_REMOVALS = "Instructing␣the␣other␣group␣
            members␣to␣expunge␣your␣documents␣from␣their␣panoramas.";
        private final static java.lang.String NO_GROUP_SELECTED_TO_EDIT_MESSAGE = "You␣must␣select␣a␣group␣
            from␣the␣list␣before␣you␣can␣edit.";
154     private final static java.lang.String GROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE = "The␣group␣cannot
            ␣be␣deleted␣because␣it␣currently␣has␣one␣or␣more␣active␣participants.";
        private final static java.util.Hashtable MEDIA_TYPES = new Hashtable();
156     private final static java.awt.Color RESIZE_RECTANGLE_PEN_COLOUR = Color.green;
        private final static java.awt.Color ZOOM_VECTOR_COLOUR = Color.green;
158     private final static int DOCUMENT_BORDER_WIDTH = 6;
        public final static java.awt.Color DOCUMENT_MATTE_BORDER_COLOUR = Color.gray;
160     private final static java.awt.Color DOCUMENT_LINE_BORDER_COLOUR = Color.lightGray;
        private final static int ZOOM_VECTOR_WIDTH = 2;
162     public final static float RENDER_CUTOFF = (float)0.1;
        private final static java.lang.String PANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE = "One␣or␣more␣of␣
            the␣files␣you␣attempted␣to␣add␣were␣already␣part␣of␣the␣panorama.";
164
        private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.9;
166     private final static java.lang.String CLIPBOARD_NAME = "Peerview␣clipboard";
        private final static java.lang.String COPIED_TEXT_MESSAGE = "Copied␣text␣to␣the␣clipboard␣-␣it␣can␣
            now␣be␣pasted␣into␣a␣message";
168     private final static java.lang.String NO_TEXT_TO_COPY_MESSAGE = "No␣text␣was␣copied␣since␣none␣was␣
            selected.";
        private final static int RESET_MESSAGE_AREA_DELAY = 7000;
170     private final static java.lang.String CONTENTS_AND_SIGNATURE_DIVIDER = "\n\n\n\n";

172     private final static java.lang.String LOCAL_MODE_MESSAGE = "Unable␣to␣connect␣to␣server␣or␣the␣most␣
            recently␣joined␣group.␣Peerview␣is␣therefore␣in␣local␣mode␣meaning␣that␣no␣information␣will␣be␣
            exchanged␣with␣neither␣the␣server␣nor␣other␣group␣members.␣To␣attempt␣a␣new␣connection,␣please␣
            join␣one␣of␣the␣groups␣listed␣in␣the␣group␣directory␣available␣from␣the␣menubar.";
        private final static java.lang.String ATTEMPT_CONNECTION_CONFIRMATION_MESSAGE = "There␣is␣currently␣
            no␣connection␣to␣the␣server.␣Should␣a␣connection␣attempt␣be␣made␣?";
174     private final static int GROUP_DIRECTORY_REQUEST_DELAY = 5000;
        private final static java.lang.String REQUESTING_GROUPS_DIRECTORY = "Requesting␣group␣directory␣from␣
            server...";
176     public final static java.lang.String FAILED_TO_RECEIVE_GROUP_DIRECTORY = "Failed␣to␣receive␣a␣group␣
            directory␣within␣the␣preset␣time␣interval.␣You␣should␣check␣server␣settings␣and␣verify␣that␣the␣
            server␣and␣network␣are␣functional␣and␣responding.␣If␣the␣problem␣is␣caused␣by␣network␣lag,␣the␣
            group␣directory␣may␣still␣arrive␣shortly.";
        private final static java.lang.String CONNECTED_TO_GROUP_MANAGEMENT = "Successfully␣connected␣to␣
            server.";
178     private final static java.lang.String FAILED_TO_CONNECT_TO_SERVER = "Could␣not␣connect␣to␣server.";
        private final static java.lang.String CONFIRM_CONNECT_TO_SERVER = "Connecting␣to␣a␣new␣server␣will␣
            disconnect␣you␣from␣the␣group␣and␣server␣you␣are␣currently␣connected␣to,␣if␣any.␣Do␣you␣want␣to␣
            continue␣?";
180     private final static java.lang.String TITLE_REQUIRED_MESSAGE = "You␣must␣provide␣a␣title␣for␣your␣
            message␣before␣it␣can␣be␣submitted.";

182     private final static java.lang.String NOTHING_TO_REMOVE_MESSAGE = "Nothing␣to␣remove.";
        private final static java.lang.String NO_GROUP_SELECTED_TO_DELETE = "You␣must␣select␣a␣group␣to␣
            delete.";
184     private final static String GROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS = "The␣group␣cannot␣be
            ␣deleted␣as␣it␣has␣one␣or␣more␣active␣participants.";
        private final static java.lang.String GROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS = "This␣group
            ␣cannot␣be␣edited␣as␣it␣has␣one␣or␣more␣active␣participants.";
186     private final static java.lang.String REMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE = "Removing␣the␣other␣
            group␣members'␣documents␣from␣the␣panorama...";
        private final static java.lang.String LAST_MODIFIED = "Last␣modified";
188     private final static java.lang.String DOCUMENT_NAME = "Document␣name";
        private final static java.lang.String RAW_BITMAP = "Raw␣bitmap";
190     private final static java.lang.String DOCUMENT_UPDATE = "Document␣update";
        private final static java.lang.String UPDATING_DOCUMENTS = "Now␣analyzing␣your␣documents␣and␣
            broadcasting␣updates␣as␣necessary␣to␣the␣other␣group␣members.";
192     private final static int ANIMATION_MAX_SPEED = 7000;
        private final static int ANIMATION_MIN_SPEED = 300;
194     private final static int UPDATE_NOTIFICATION_THRESHOLD = 15;
```

```
    private final static java.lang.String VISIBLE_DOCUMENTS_BOX_CAPTION = "Documents␣in␣panorama";
196 private final static int VISIBLE_DOCUMENTS_ICON_TEXT_GAP = 8;
    private final static java.lang.String VISIBLE_DOCUMENTS_TAB_SIZE = "␣␣␣␣␣␣";
198 private final static java.lang.String SENDER_NAME = "Sender␣name";

200 private final static java.lang.String FINISHED_ADDING_DOCUMENTS = "Finished␣adding␣documents";
    private final static java.lang.String UPDATE_PANORAMA = "Update␣panorama";
202 private final static java.awt.Color VISIBLE_DOCUMENTS_SELECTION_COLOUR = new Color(0,0,80);
    private final static java.lang.String UPDATE_PANORAMA_BY_REQUEST_MESSAGE = "Updating␣the␣panorama␣by␣
        request␣from␣other␣group␣member.";
204 private final static java.awt.Dimension VISIBLE_DOCUMENTS_ITEM_MAX_SIZE = new java.awt.Dimension
        ( 350, 30 );
    private final static java.lang.String ABOUT_BOX_MESSAGE = "Release␣0.900␣(beta).";
206 private final static int MAX_GRID_LAYOUT_DOCUMENT_HEIGHT = 2000;
    private final static int MIN_GRID_LAYOUT_DOCUMENT_HEIGHT = 300;
208 private final static int MAX_GRID_LAYOUT_DOCUMENT_WIDTH = 1500;
    private final static int MIN_GRID_LAYOUT_DOCUMENT_WIDTH = 400;
210 private final static int MAX_GRID_LAYOUT_HORIZONTAL_SPACING = 500;
    private final static int MIN_GRID_LAYOUT_HORIZONTAL_SPACING = 0;
212 private final static int MIN_GRID_LAYOUT_VERTICAL_SPACING = 0;
    private final static int MAX_GRID_LAYOUT_VERTICAL_SPACING = 500;
214 private final static java.lang.String DOCUMENT_LIST_FILENAME = "doclist.dat";
    private final static java.lang.String COULD_NOT_READ_DOCUMENT_LIST = "Unable␣to␣read␣and␣process␣the␣
        list␣of␣documents␣from␣disk.";
216 private final static java.lang.String COULD_NOT_WRITE_DOCUMENT_LIST = "Could␣not␣write␣the␣list␣of␣
        documents␣to␣disk.";
    private final static int TEXT_DOCUMENT_THRESHOLD = 1000000;
218 private final static java.lang.String NOT_CONNECTED_TO_SERVER = "This␣action␣cannot␣be␣completed␣
        because␣there␣is␣currently␣no␣connection␣to␣the␣server.";
    private final static int MAX_DOCNAME_LENGTH = 75;
220 private final static java.lang.String NODE_LOCKED = "Node␣locked";
    private final static java.lang.String AUTOLOCK_NODES = "Autolock␣nodes";
222 private final static java.lang.String MESSAGE_LOG_FILENAME = "Message␣log␣filename";
    private final static java.lang.String MESSAGE_LOG_WRITE_ERROR = "Could␣not␣write␣the␣message␣log␣to␣
        disk.";
224 private final static java.lang.String MESSAGE_LOG_MESSAGE = "Wrote␣message␣log␣to␣disk";
    private final static java.lang.String NOTIFYING_SERVER_OF_NAME_IN_USE = "Your␣access␣to␣the␣server␣is
        ␣blocked.␣Requesting␣a␣clean-up...";
226 private final static java.lang.String CLIENT_DOCUMENTATION_HELPSET_NAME = "peerview.hs";
    private final static java.lang.String COULD_NOT_INITIALIZE_HELP = "Could␣not␣initialize␣the␣help␣
        system.␣Online␣help␣will␣not␣be␣available.";
228 private final static java.lang.String CLIENT_HELP_PATH = "help";
    private static java.util.Hashtable FILE_TYPE_SYNONYMS = null;
230
  /**
232 * ClientMessages constructor comment.
    */
234 public ClientConstants() {
    super();
236 initialize();
  }
238 /**
    * Insert the method's description here.
240 * Creation date: (15-06-00 23:12:13)
    * @return java.lang.String
242 * @param fileName java.lang.String
    */
244 static public String failedToAccessDocument(String fileName) {
    return "An␣error␣occurred␣while␣trying␣to␣access␣the␣document␣with␣filename␣\"" + fileName + "\".";
246 }
  /**
248 * Insert the method's description here.
    * Creation date: (02-09-00 12:29:51)
250 * @return java.lang.String
    */
252 public final static java.lang.String getABOUT_BOX_MESSAGE() {
```

```
        return ABOUT_BOX_MESSAGE;
254  }
     /**
256   * Insert the method's description here.
      * Creation date: (09-08-00 23:05:17)
258   * @return java.lang.String
      */
260  public final static java.lang.String getACTIVE_GROUP_UPON_EXIT() {
        return ACTIVE_GROUP_UPON_EXIT;
262  }
     /**
264   * Insert the method's description here.
      * Creation date: (19-08-00 20:59:44)
266   * @param documentName java.lang.String
      * @param senderName java.lang.String
268   */
     public final static String getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(String documentName, String senderName)
270  {
        return "Received␣" + documentName + "␣from␣" + senderName;
272  }
     /**
274   * Insert the method's description here.
      * Creation date: (05-09-00 14:50:29)
276   * @return java.lang.String
      * @param authorName java.lang.String
278   * @param numDoks int
      */
280  public final static String getADDING_DISTRIBUTED_DOCUMENTS(String authorName, int numDoks)
     {
282     return "Adding␣" + numDoks + "␣documents␣received␣from␣" + authorName;
     }
284  /**
      * Insert the method's description here.
286   * Creation date: (05-10-00 23:05:28)
      * @return java.lang.String
288   */
     public final static java.lang.String getALL_FILES() {
290     return ALL_FILES;
     }
292  /**
      * Insert the method's description here.
294   * Creation date: (25-07-00 15:52:24)
      * @return java.lang.String
296   */
     public final static java.lang.String getALREADY_JOINED_TO_GROUP_MESSAGE() {
298     return ALREADY_JOINED_TO_GROUP_MESSAGE;
     }
300  /**
      * Insert the method's description here.
302   * Creation date: (13-07-00 13:08:57)
      * @return java.lang.String
304   */
     public final static java.lang.String getANIMATION_DURATION_IN_MILLISECS() {
306     return ANIMATION_DURATION_IN_MILLISECS;
     }
308  /**
      * Insert the method's description here.
310   * Creation date: (22-08-00 17:53:01)
      * @return double
312   */
     public final static int getANIMATION_MAX_SPEED() {
314     return ANIMATION_MAX_SPEED;
     }
316  /**
      * Insert the method's description here.
318   * Creation date: (22-08-00 17:56:11)
```

```
     * @return double
320   */
     public final static int getANIMATION_MIN_SPEED() {
322     return ANIMATION_MIN_SPEED;
     }
324  /**
     * Insert the method's description here.
326  * Creation date: (10-08-00 15:38:52)
     * @return java.lang.String
328  */
     public final static java.lang.String getATTEMPT_CONNECTION_CONFIRMATION_MESSAGE() {
330     return ATTEMPT_CONNECTION_CONFIRMATION_MESSAGE;
     }
332  /**
     * Insert the method's description here.
334  * Creation date: (17-07-00 13:52:34)
     * @return java.lang.String
336  */
     public final static java.lang.String getAUTHOR_NAME() {
338     return AUTHOR_NAME;
     }
340  /**
     * Insert the method's description here.
342  * Creation date: (14-10-00 11:31:51)
     * @return java.lang.String
344  */
     public final static java.lang.String getAUTOLOCK_NODES() {
346     return AUTOLOCK_NODES;
     }
348  /**
     * Insert the method's description here.
350  * Creation date: (16-06-00 17:55:28)
     * @return int
352  */
     public final static Color getBACKGROUND_COLOUR() {
354     return BACKGROUND_COLOUR;
     }
356  /**
     * Insert the method's description here.
358  * Creation date: (22-06-00 20:22:01)
     * @return java.lang.String
360  */
     public final static java.lang.String getBITMAP() {
362     return BITMAP;
     }
364  /**
     * Insert the method's description here.
366  * Creation date: (26-07-00 14:29:56)
     * @return java.lang.String
368  */
     public final static java.lang.String getBROADCASTING_DOCUMENT_REMOVALS() {
370     return BROADCASTING_DOCUMENT_REMOVALS;
     }
372  /**
     * Insert the method's description here.
374  * Creation date: (08-07-00 19:00:37)
     * @return java.lang.String
376  * @param numberOfDocuments int
     * @param numberOfRecipients int
378  */
     public static String getBROADCASTING_DOCUMENTS_MESSAGE(int numberOfDocuments) {
380     return "Broadcasting␣" + String.valueOf( numberOfDocuments )+ "␣document(s)";
     }
382  /**
     * Insert the method's description here.
384  * Creation date: (08-07-00 19:00:37)
```

```
     * @return java.lang.String
386  * @param numberOfDocuments int
     * @param numberOfRecipients int
388  */
     public static String getBROADCASTING_DOCUMENTS_MESSAGE(int numberOfDocuments, int numberOfRecipients) {
390     return "Broadcasting␣to␣other␣group␣members:␣" + String.valueOf( numberOfDocuments )+ "␣document(s)␣
            to␣" + String.valueOf( numberOfRecipients )+ "␣recipient(s).";
     }
392  /**
     * Insert the method's description here.
394  * Creation date: (15-11-00 19:42:53)
     * @return java.lang.String
396  */
     public final static java.lang.String getCLIENT_DOCUMENTATION_HELPSET_NAME() {
398     return CLIENT_DOCUMENTATION_HELPSET_NAME;
     }
400  /**
     * Insert the method's description here.
402  * Creation date: (15-11-00 23:08:27)
     * @return java.lang.String
404  */
     public final static java.lang.String getCLIENT_HELP_PATH() {
406     return CLIENT_HELP_PATH;
     }
408  /**
     * Insert the method's description here.
410  * Creation date: (26-07-00 12:37:23)
     * @return java.lang.String
412  * @param workGroup peerviewmisc.WorkGroup
     */
414  public static String getCLIENT_JOINED_GROUP_MESSAGE(peerviewmisc.WorkGroup workGroup) {
        return "You␣have␣now␣joined␣" + workGroup.getGroupName() + ".";
416  }
     /**
418  * Insert the method's description here.
     * Creation date: (21-08-00 08:39:23)
420  * @return java.lang.String
     * @param workGroup peerviewmisc.WorkGroup
422  */
     public final static String getCLIENT_JOINED_GROUP_WITH_MEMBERS_MESSAGE(peerviewmisc.WorkGroup workGroup)
           {
424     return "You␣have␣now␣joined␣" + workGroup.getGroupName() + ".␣" + "The␣other␣members␣have␣been␣
            notified␣and␣should␣now␣be␣submitting␣their␣documents,␣if␣any,␣to␣you.";

426  }
     /**
428  * Insert the method's description here.
     * Creation date: (26-07-00 03:10:56)
430  * @param clientsAuthorName java.lang.String
     */
432  public static String getCLIENT_LEFT_GROUP_MESSAGE(String clientsAuthorName)
     {
434     return clientsAuthorName + "␣has␣left␣the␣group␣and␣his/her␣documents␣have␣been␣removed␣from␣your␣
            panorama.";
     }
436  /**
     * Insert the method's description here.
438  * Creation date: (07-08-00 16:00:41)
     * @return java.lang.String
440  */
     public final static java.lang.String getCLIPBOARD_NAME() {
442     return CLIPBOARD_NAME;
     }
444  /**
     * Insert the method's description here.
446  * Creation date: (10-08-00 21:38:21)
```

```
     * @return java.lang.String
448   */
     public final static java.lang.String getCONFIRM_CONNECT_TO_SERVER() {
450     return CONFIRM_CONNECT_TO_SERVER;
     }
452  /**
     * Insert the method's description here.
454  * Creation date: (12-07-00 11:09:48)
     * @return java.lang.String
456  * @param messageID java.lang.String
     */
458  public static String getCONFIRM_MESSAGE_DELETION(String messageID)
     {
460     return "Delete␣" + messageID + "␣?";
     }
462  /**
     * Insert the method's description here.
464  * Creation date: (26-07-00 20:41:52)
     * @param groupName java.lang.String
466  */
     public static String getCONFIRM_WORKGROUP_DELETION(String groupName)
468  {
       return "Delete␣the␣group:␣" + "\"" + groupName + "\"␣?";
470  }
     /**
472  * Insert the method's description here.
     * Creation date: (10-08-00 20:39:43)
474  * @return java.lang.String
     */
476  public final static java.lang.String getCONNECTED_TO_GROUP_MANAGEMENT() {
       return CONNECTED_TO_GROUP_MANAGEMENT;
478  }
     /**
480  * Insert the method's description here.
     * Creation date: (09-10-00 12:57:14)
482  * @return java.lang.String
     * @param connectionEventString java.lang.String
484  */
     public static final String getCONNECTION_FAILURE(String connectionEventString)
486  {
       return "Connection␣failure␣-␣communication␣with␣the␣server␣has␣been␣disrupted␣(␣" +
            connectionEventString + ")";
488  }
     /**
490  * Insert the method's description here.
     * Creation date: (08-08-00 10:28:46)
492  * @return java.lang.String
     */
494  public final static java.lang.String getCONTENTS_AND_SIGNATURE_DIVIDER() {
       return CONTENTS_AND_SIGNATURE_DIVIDER;
496  }
     /**
498  * Insert the method's description here.
     * Creation date: (07-08-00 17:36:30)
500  * @return java.lang.String
     */
502  public final static java.lang.String getCOPIED_TEXT_MESSAGE() {
       return COPIED_TEXT_MESSAGE;
504  }
     /**
506  * Insert the method's description here.
     * Creation date: (10-08-00 02:27:09)
508  * @return java.lang.String
     * @param groupName java.lang.String
510  */
     public static String getCOULD_NOT_CONNECT_TO_GROUP_MESSAGE(String groupName)
```

```
512  {
        return "Could␣not␣establish␣a␣connection␣to␣the␣group␣\"" + groupName + "\"";
514  }
     /**
516   * Insert the method's description here.
      * Creation date: (15-11-00 19:46:55)
518   * @return java.lang.String
      */
520  public final static java.lang.String getCOULD_NOT_INITIALIZE_HELP() {
        return COULD_NOT_INITIALIZE_HELP;
522  }
     /**
524   * Insert the method's description here.
      * Creation date: (06-09-00 21:45:12)
526   * @return java.lang.String
      */
528  public final static java.lang.String getCOULD_NOT_READ_DOCUMENT_LIST() {
        return COULD_NOT_READ_DOCUMENT_LIST;
530  }
     /**
532   * Insert the method's description here.
      * Creation date: (06-09-00 22:04:19)
534   * @return java.lang.String
      */
536  public final static java.lang.String getCOULD_NOT_WRITE_DOCUMENT_LIST() {
        return COULD_NOT_WRITE_DOCUMENT_LIST;
538  }
     /**
540   * Insert the method's description here.
      * Creation date: (19-07-00 10:48:58)
542   * @return java.lang.String
      */
544  public final static java.lang.String getDEFAULT_DOCUMENT_HEIGHT() {
        return DEFAULT_DOCUMENT_HEIGHT;
546  }
     /**
548   * Insert the method's description here.
      * Creation date: (19-07-00 10:47:47)
550   * @return java.lang.String
      */
552  public final static java.lang.String getDEFAULT_DOCUMENT_WIDTH() {
        return DEFAULT_DOCUMENT_WIDTH;
554  }
     /**
556   * Insert the method's description here.
      * Creation date: (03-08-00 17:21:36)
558   * @return java.lang.String
      */
560  public final static java.lang.String getDEFAULT_FRAME_HORIZONTAL_SIZE() {
        return DEFAULT_FRAME_HORIZONTAL_SIZE;
562  }
     /**
564   * Insert the method's description here.
      * Creation date: (03-08-00 17:21:54)
566   * @return java.lang.String
      */
568  public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
        return DEFAULT_FRAME_VERTICAL_SIZE;
570  }
     /**
572   * Insert the method's description here.
      * Creation date: (03-08-00 18:10:49)
574   * @return java.lang.String
      */
576  public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
        return DEFAULT_FRAME_X_COORDINATE;
```

```
578  }
     /**
580   * Insert the method's description here.
      * Creation date: (03-08-00 18:11:11)
582   * @return java.lang.String
      */
584  public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
        return DEFAULT_FRAME_Y_COORDINATE;
586  }
     /**
588   * Insert the method's description here.
      * Creation date: (06-09-00 00:48:38)
590   * @return java.lang.String
      */
592  public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT()
     {
594     return DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT;
     }
596  /**
      * Insert the method's description here.
598   * Creation date: (06-09-00 00:47:52)
      * @return java.lang.String
600   */
     public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH() {
602     return DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH;
     }
604  /**
      * Insert the method's description here.
606   * Creation date: (06-09-00 00:50:09)
      * @return java.lang.String
608   */
     public final static java.lang.String getDEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING() {
610     return DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING;
     }
612  /**
      * Insert the method's description here.
614   * Creation date: (06-09-00 00:50:42)
      * @return java.lang.String
616   */
     public final static java.lang.String getDEFAULT_GRID_LAYOUT_VERTICAL_SPACING() {
618     return DEFAULT_GRID_LAYOUT_VERTICAL_SPACING;
     }
620  /**
      * Insert the method's description here.
622   * Creation date: (11-08-00 17:12:57)
      * @return java.lang.String
624   */
     public final static java.lang.String getDEFAULT_GROUP_ID() {
626     return DEFAULT_GROUP_ID;
     }
628  /**
      * Insert the method's description here.
630   * Creation date: (19-07-00 10:52:34)
      * @return java.lang.String
632   */
     public final static java.lang.String getDEFAULT_HORIZONTAL_DOCUMENT_SPACING() {
634     return DEFAULT_HORIZONTAL_DOCUMENT_SPACING;
     }
636  /**
      * Insert the method's description here.
638   * Creation date: (28-06-00 13:37:45)
      * @return double
640   */
     public final static double getDEFAULT_PANORAMA_PERCENTUAL_SIZE() {
642     return DEFAULT_PANORAMA_PERCENTUAL_SIZE;
     }
```

```
644  /**
      * Insert the method's description here.
646   * Creation date: (25-06-00 12:17:08)
      * @return java.lang.String[][]
648   */
     public final static Properties getDEFAULT_PREFERENCES()
650  {
        Properties p = new Properties();
652
        for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
654     {
           p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
656     }
        return p;
658  }
     /**
660   * Insert the method's description here.
      * Creation date: (19-07-00 10:53:28)
662   * @return java.lang.String
      */
664  public final static java.lang.String getDEFAULT_VERTICAL_DOCUMENT_SPACING() {
        return DEFAULT_VERTICAL_DOCUMENT_SPACING;
666  }
     /**
668   * Insert the method's description here.
      * Creation date: (25-06-00 20:55:23)
670   * @return java.lang.String[][]
      */
672  public final static Hashtable getDISPLAY_QUALITIES()
     {
674     Hashtable displayQualities = new Hashtable();
        for ( int i = 0 ; i < DISPLAY_QUALITIES.length; i++ )
676     {
           displayQualities.put( DISPLAY_QUALITIES[i][0], DISPLAY_QUALITIES[i][1] );
678     }
        return displayQualities;
680  }
     /**
682   * Insert the method's description here.
      * Creation date: (25-06-00 16:36:00)
684   * @return java.lang.String
      */
686  public final static java.lang.String getDISPLAY_QUALITY() {
        return DISPLAY_QUALITY;
688  }
     /**
690   * Insert the method's description here.
      * Creation date: (08-07-00 13:59:10)
692   * @return java.lang.String
      */
694  public final static java.lang.String getDISTRIBUTED_DOCUMENT() {
        return DISTRIBUTED_DOCUMENT;
696  }
     /**
698   * Insert the method's description here.
      * Creation date: (15-08-00 17:50:16)
700   * @return java.lang.String
      */
702  public final static java.lang.String getDO_NOT_ZOOM_TO_OVERVIEW() {
        return DO_NOT_ZOOM_TO_OVERVIEW;
704  }
     /**
706   * Insert the method's description here.
      * Creation date: (30-07-00 20:40:10)
708   * @return int
      */
```

```
public final static int getDOCUMENT_BORDER_WIDTH() {
   return DOCUMENT_BORDER_WIDTH;
}
/**
 * Insert the method's description here.
 * Creation date: (22-08-00 01:26:41)
 * @return java.lang.String
 * @param documentName java.lang.String
 * @param authorName java.lang.String
 * @param size long
 * @param lastUpdate java.util.Date
 */
public final static String getDOCUMENT_LABEL(String documentName, String authorName, long size, java.
     util.Date lastUpdate)
{ java.util.GregorianCalendar calendar = new java.util.GregorianCalendar();
   calendar.setTime( lastUpdate );
   return  "Name:" + documentName + "\n" +
         "Author:␣" + authorName + "\n" +
         "Size:␣" + String.valueOf( size )+ "␣bytes" + "\n" +
         "Last␣updated:␣" + (new java.text.SimpleDateFormat()).format( lastUpdate )+ ":" + String.valueOf
              ( calendar.get( java.util.Calendar.SECOND ));
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 20:49:44)
 * @return java.awt.Color
 */
public final static java.awt.Color getDOCUMENT_LINE_BORDER_COLOUR() {
   return DOCUMENT_LINE_BORDER_COLOUR;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 21:36:21)
 * @return java.lang.String
 */
public final static java.lang.String getDOCUMENT_LIST_FILENAME()
{
   return DOCUMENT_LIST_FILENAME;
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 20:44:17)
 * @return java.awt.Color
 */
public final static java.awt.Color getDOCUMENT_MATTE_BORDER_COLOUR() {
   return DOCUMENT_MATTE_BORDER_COLOUR;
}
/**
 * Insert the method's description here.
 * Creation date: (21-08-00 12:34:32)
 * @return java.lang.String
 */
public final static java.lang.String getDOCUMENT_NAME() {
   return DOCUMENT_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (09-07-00 17:44:32)
 * @return java.lang.String
 */
public final static java.lang.String getDOCUMENT_PATH() {
   return DOCUMENT_PATH;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 14:52:54)
```

```
774   * @return java.lang.String
      */
776 public static final String getDOCUMENT_SIZE_OUT_OF_BOUNDS()
    {
778    return "Documents␣must␣be␣between␣" + String.valueOf( getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() )+ "␣and␣"
             + String.valueOf( getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() )+ "␣in␣width␣and␣between␣"
780          + String.valueOf( getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() )+ "␣and␣"
             + String.valueOf( getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() )+ "␣in␣height.";
782 }
    /**
784  * Insert the method's description here.
     * Creation date: (21-08-00 16:54:07)
786  * @return java.lang.String
     */
788 public final static java.lang.String getDOCUMENT_UPDATE() {
       return DOCUMENT_UPDATE;
790 }
    /**
792  * Insert the method's description here.
     * Creation date: (10-08-00 20:46:02)
794  * @return java.lang.String
     */
796 public final static java.lang.String getFAILED_TO_CONNECT_TO_SERVER() {
       return FAILED_TO_CONNECT_TO_SERVER;
798 }
    /**
800  * Insert the method's description here.
     * Creation date: (10-08-00 16:02:45)
802  * @return java.lang.String
     */
804 public final static java.lang.String getFAILED_TO_RECEIVE_GROUP_DIRECTORY() {
       return FAILED_TO_RECEIVE_GROUP_DIRECTORY;
806 }
    /**
808  * Insert the method's description here.
     * Creation date: (30-08-00 17:27:03)
810  * @return java.lang.String
     */
812 public final static java.lang.String getFALSE() {
       return FALSE;
814 }
    /**
816  * Insert the method's description here.
     * Creation date: (22-06-00 19:28:09)
818  * @return int
     */
820 public final static int getFILE_EXTENSION_INDEX() {
       return FILE_EXTENSION_INDEX;
822 }
    /**
824  * Insert the method's description here.
     * Creation date: (22-06-00 19:28:32)
826  * @return int
     */
828 public final static int getFILE_TYPE_DESCRIPTION_INDEX() {
       return FILE_TYPE_DESCRIPTION_INDEX;
830 }
    /**
832  * Insert the method's description here.
     * Creation date: (23-02-01 08:49:05)
834  * @return java.lang.String[][]
     */
836 public final static java.lang.String[][] getFILE_TYPE_SYNONYM_TABLE() {
       return FILE_TYPE_SYNONYM_TABLE;
838 }
    /**
```

```
840   * Insert the method's description here.
      * Creation date: (23-02-01 08:42:21)
842   * @return java.util.Hashtable
      */
844  public static java.util.Hashtable getFILE_TYPE_SYNONYMS()
     {
846     if ( FILE_TYPE_SYNONYMS == null )
        {
848        FILE_TYPE_SYNONYMS = new Hashtable();
           for ( int i = 0; i < getFILE_TYPE_SYNONYM_TABLE().length; i++ )
850        {
              java.util.ArrayList synonymList = new java.util.ArrayList();
852           for (int j = 0; j < getFILE_TYPE_SYNONYM_TABLE()[i].length; j++ )
              {
854              synonymList.add( getFILE_TYPE_SYNONYM_TABLE()[i][j] );
              }
856           FILE_TYPE_SYNONYMS.put( getFILE_TYPE_SYNONYM_TABLE()[i][0], synonymList );
           }
858
        }
860     return FILE_TYPE_SYNONYMS;
     }
862  /**
      * Insert the method's description here.
864   * Creation date: (20-06-00 08:23:58)
      * @return java.lang.String[]
866   */
     public final static java.lang.String[][] getFileTypes() {
868     return FileTypes;
     }
870  /**
      * Insert the method's description here.
872   * Creation date: (30-08-00 22:01:04)
      * @return java.lang.String
874   */
     public final static java.lang.String getFINISHED_ADDING_DOCUMENTS() {
876     return FINISHED_ADDING_DOCUMENTS;
     }
878  /**
      * Insert the method's description here.
880   * Creation date: (03-08-00 17:20:38)
      * @return java.lang.String
882   */
     public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
884     return FRAME_HORIZONTAL_SIZE;
     }
886  /**
      * Insert the method's description here.
888   * Creation date: (05-08-00 20:11:48)
      * @return double
890   */
     public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
892     return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
     }
894  /**
      * Insert the method's description here.
896   * Creation date: (03-08-00 17:21:19)
      * @return java.lang.String
898   */
     public final static java.lang.String getFRAME_VERTICAL_SIZE() {
900     return FRAME_VERTICAL_SIZE;
     }
902  /**
      * Insert the method's description here.
904   * Creation date: (03-08-00 18:09:16)
      * @return java.lang.String
```

```
 */
public final static java.lang.String getFRAME_X_COORDINATE() {
   return FRAME_X_COORDINATE;
}
/**
 * Insert the method's description here.
 * Creation date: (03-08-00 18:09:58)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_Y_COORDINATE() {
   return FRAME_Y_COORDINATE;
}
/**
 * Insert the method's description here.
 * Creation date: (25-06-00 13:32:26)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT() {
   return GRID_LAYOUT;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 13:00:42)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() {
   return GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 13:00:07)
 * @return java.lang.String
 */
public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() {
   return GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS;
}
/**
 * Insert the method's description here.
 * Creation date: (06-09-00 15:05:25)
 * @return java.lang.String
 */
public final static String getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS()
{
   return  "Horizontal␣document␣spacing␣must␣be␣between␣" +
         getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() + "␣and␣" +
         getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() + "." +
         "Vertical␣document␣spacing␣must␣be␣between␣" +
         getMIN_GRID_LAYOUT_VERTICAL_SPACING() + "␣and␣" +
         getMAX_GRID_LAYOUT_VERTICAL_SPACING() + ".";
}
/**
 * Insert the method's description here.
 * Creation date: (25-07-00 14:56:17)
 * @return java.lang.String
 */
public final static java.lang.String getGROUP_AT_MAXIMUM_MEMBERS_MESSAGE() {
   return GROUP_AT_MAXIMUM_MEMBERS_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (18-08-00 02:02:33)
 * @return int
 */
public final static String getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() {
   return GROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS;
}
```

```
972  /**
      * Insert the method's description here.
974   * Creation date: (18-08-00 02:06:15)
      * @return java.lang.String
976   */
     public final static java.lang.String getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() {
978     return GROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS;
     }
980  /**
      * Insert the method's description here.
982   * Creation date: (21-07-00 12:48:25)
      * @return int
984   */
     public final static int getGROUP_DIRECTORY_COLUMN_WIDTH() {
986     return GROUP_DIRECTORY_COLUMN_WIDTH;
     }
988  /**
      * Insert the method's description here.
990   * Creation date: (10-08-00 15:39:59)
      * @return int
992   */
     public final static int getGROUP_DIRECTORY_REQUEST_DELAY() {
994     return GROUP_DIRECTORY_REQUEST_DELAY;
     }
996  /**
      * Insert the method's description here.
998   * Creation date: (26-07-00 20:32:52)
      * @return java.lang.String
1000  */
     public final static java.lang.String getGROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE() {
1002    return GROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE;
     }
1004 /**
      * Insert the method's description here.
1006  * Creation date: (22-06-00 20:20:34)
      * @return java.lang.String
1008  */
     public final static java.lang.String getHTML() {
1010    return HTML;
     }
1012 /**
      * Insert the method's description here.
1014  * Creation date: (18-07-00 11:23:18)
      * @return java.lang.String
1016  */
     public final static java.lang.String getINVALID_DELETION_ATTEMPTED_MESSAGE() {
1018    return INVALID_DELETION_ATTEMPTED_MESSAGE;
     }
1020 /**
      * Insert the method's description here.
1022  * Creation date: (21-08-00 12:26:41)
      * @return java.lang.String
1024  */
     public final static java.lang.String getLAST_MODIFIED() {
1026    return LAST_MODIFIED;
     }
1028 /**
      * Insert the method's description here.
1030  * Creation date: (25-06-00 16:38:21)
      * @return java.lang.String
1032  */
     public final static java.lang.String getLAYOUT_SCHEME() {
1034    return LAYOUT_SCHEME;
     }
1036 /**
      * Insert the method's description here.
```

```
1038  * Creation date: (25-06-00 21:32:43)
      * @return java.lang.String[][]
1040  */
      public final static java.lang.String[] getLAYOUT_SCHEMES() {
1042     return LAYOUT_SCHEMES;
      }
1044  /**
      * Insert the method's description here.
1046  * Creation date: (06-09-00 20:36:54)
      * @return java.lang.String
1048  */
      public final static java.lang.String getLOAD_DOCUMENT_LIST_ON_STARTUP() {
1050     return LOAD_DOCUMENT_LIST_ON_STARTUP;
      }
1052  /**
      * Insert the method's description here.
1054  * Creation date: (07-07-00 23:09:39)
      * @return int
1056  */
      public final static int getLOCAL_DOCUMENT() {
1058     return LOCAL_DOCUMENT;
      }
1060  /**
      * Insert the method's description here.
1062  * Creation date: (26-06-00 09:55:24)
      * @return java.lang.String
1064  */
      public final static java.lang.String getLOCAL_HOST() {
1066     return LOCAL_HOST;
      }
1068  /**
      * Insert the method's description here.
1070  * Creation date: (10-08-00 04:40:26)
      * @return java.lang.String
1072  */
      public final static java.lang.String getLOCAL_MODE_MESSAGE() {
1074     return LOCAL_MODE_MESSAGE;
      }
1076  /**
      * Insert the method's description here.
1078  * Creation date: (10-10-00 00:09:18)
      * @return int
1080  */
      public final static int getMAX_DOCNAME_LENGTH() {
1082     return MAX_DOCNAME_LENGTH;
      }
1084  /**
      * Insert the method's description here.
1086  * Creation date: (06-09-00 14:48:29)
      * @return int
1088  */
      public final static int getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() {
1090     return MAX_GRID_LAYOUT_DOCUMENT_HEIGHT;
      }
1092  /**
      * Insert the method's description here.
1094  * Creation date: (06-09-00 14:50:48)
      * @return int
1096  */
      public final static int getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() {
1098     return MAX_GRID_LAYOUT_DOCUMENT_WIDTH;
      }
1100  /**
      * Insert the method's description here.
1102  * Creation date: (06-09-00 14:58:01)
      * @return int
```

```
1104   */
       public final static int getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() {
1106      return MAX_GRID_LAYOUT_HORIZONTAL_SPACING;
       }
1108   /**
        * Insert the method's description here.
1110    * Creation date: (06-09-00 15:02:56)
        * @return int
1112    */
       public final static int getMAX_GRID_LAYOUT_VERTICAL_SPACING() {
1114      return MAX_GRID_LAYOUT_VERTICAL_SPACING;
       }
1116   /**
        * Insert the method's description here.
1118    * Creation date: (27-07-00 12:35:59)
        * @return java.util.Hashtable
1120    */
       public final static java.util.Hashtable getMEDIA_TYPES()
1122   {
          if ( MEDIA_TYPES.size() == 0 )
1124      {
             for ( int i = 0; i < getFileTypes().length; i++ )
1126         {
                MEDIA_TYPES.put( getFileTypes()[i][0], getFileTypes()[i][2] );
1128         }
          }
1130      return MEDIA_TYPES;
       }
1132   /**
        * Insert the method's description here.
1134    * Creation date: (18-06-00 17:33:32)
        * @return java.lang.String
1136    */
       public final static java.lang.String getMESSAGE_AREA_DEFAULT_STRING() {
1138      return MESSAGE_AREA_DEFAULT_STRING;
       }
1140   /**
        * Insert the method's description here.
1142    * Creation date: (29-10-00 16:42:19)
        * @return java.lang.String
1144    */
       public final static java.lang.String getMESSAGE_LOG_FILENAME() {
1146      return MESSAGE_LOG_FILENAME;
       }
1148   /**
        * Insert the method's description here.
1150    * Creation date: (29-10-00 16:54:58)
        * @return java.lang.String
1152    */
       public final static java.lang.String getMESSAGE_LOG_MESSAGE() {
1154      return MESSAGE_LOG_MESSAGE;
       }
1156   /**
        * Insert the method's description here.
1158    * Creation date: (29-10-00 16:43:20)
        * @return java.lang.String
1160    */
       public final static java.lang.String getMESSAGE_LOG_WRITE_ERROR() {
1162      return MESSAGE_LOG_WRITE_ERROR;
       }
1164   /**
        * Insert the method's description here.
1166    * Creation date: (19-07-00 20:06:48)
        * @return java.lang.String
1168    */
       public final static java.lang.String getMESSAGE_PROPERTIES_NAME_COLUMN_HEADING() {
```

```
1170      return MESSAGE_PROPERTIES_NAME_COLUMN_HEADING;
      }
1172  /**
       * Insert the method's description here.
1174   * Creation date: (19-07-00 20:07:10)
       * @return java.lang.String
1176   */
      public final static java.lang.String getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADING() {
1178      return MESSAGE_PROPERTIES_VALUE_COLUMN_HEADING;
      }
1180  /**
       * Insert the method's description here.
1182   * Creation date: (06-09-00 14:49:17)
       * @return int
1184   */
      public final static int getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() {
1186      return MIN_GRID_LAYOUT_DOCUMENT_HEIGHT;
      }
1188  /**
       * Insert the method's description here.
1190   * Creation date: (06-09-00 14:51:49)
       * @return int
1192   */
      public final static int getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() {
1194      return MIN_GRID_LAYOUT_DOCUMENT_WIDTH;
      }
1196  /**
       * Insert the method's description here.
1198   * Creation date: (06-09-00 15:00:54)
       * @return int
1200   */
      public final static int getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() {
1202      return MIN_GRID_LAYOUT_HORIZONTAL_SPACING;
      }
1204  /**
       * Insert the method's description here.
1206   * Creation date: (06-09-00 15:02:09)
       * @return int
1208   */
      public final static int getMIN_GRID_LAYOUT_VERTICAL_SPACING() {
1210      return MIN_GRID_LAYOUT_VERTICAL_SPACING;
      }
1212  /**
       * Insert the method's description here.
1214   * Creation date: (25-06-00 16:33:43)
       * @return java.lang.String
1216   */
      public final static java.lang.String getNAME() {
1218      return NAME;
      }
1220  /**
       * Insert the method's description here.
1222   * Creation date: (13-08-00 19:55:27)
       * @return java.lang.String
1224   */
      public final static java.lang.String getNO_GROUP_SELECTED_TO_DELETE() {
1226      return NO_GROUP_SELECTED_TO_DELETE;
      }
1228  /**
       * Insert the method's description here.
1230   * Creation date: (26-07-00 18:23:40)
       * @return java.lang.String
1232   */
      public final static java.lang.String getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() {
1234      return NO_GROUP_SELECTED_TO_EDIT_MESSAGE;
      }
```

```
1236  /**
       * Insert the method's description here.
1238   * Creation date: (25-07-00 14:37:02)
       * @return java.lang.String
1240   */
      public final static java.lang.String getNO_GROUP_SELECTED_TO_JOIN_MESSAGE() {
1242     return NO_GROUP_SELECTED_TO_JOIN_MESSAGE;
      }
1244  /**
       * Insert the method's description here.
1246   * Creation date: (07-08-00 17:37:35)
       * @return java.lang.String
1248   */
      public final static java.lang.String getNO_TEXT_TO_COPY_MESSAGE() {
1250     return NO_TEXT_TO_COPY_MESSAGE;
      }
1252  /**
       * Insert the method's description here.
1254   * Creation date: (10-07-00 22:44:06)
       * @return java.lang.String
1256   */
      public final static java.lang.String getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() {
1258     return NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE;
      }
1260  /**
       * Insert the method's description here.
1262   * Creation date: (14-10-00 11:11:14)
       * @return java.lang.String
1264   */
      public final static java.lang.String getNODE_LOCKED() {
1266     return NODE_LOCKED;
      }
1268  /**
       * Insert the method's description here.
1270   * Creation date: (09-10-00 19:20:44)
       * @return java.lang.String
1272   */
      public final static java.lang.String getNOT_CONNECTED_TO_SERVER() {
1274     return NOT_CONNECTED_TO_SERVER;
      }
1276  /**
       * Insert the method's description here.
1278   * Creation date: (11-08-00 18:17:01)
       * @return java.lang.String
1280   */
      public final static java.lang.String getNOTHING_TO_REMOVE_MESSAGE() {
1282     return NOTHING_TO_REMOVE_MESSAGE;
      }
1284  /**
       * Insert the method's description here.
1286   * Creation date: (02-11-00 00:08:36)
       * @return java.lang.String
1288   */
      public final static java.lang.String getNOTIFYING_SERVER_OF_NAME_IN_USE() {
1290     return NOTIFYING_SERVER_OF_NAME_IN_USE;
      }
1292  /**
       * Insert the method's description here.
1294   * Creation date: (22-06-00 12:41:21)
       * @return java.lang.String
1296   */
      public final static java.lang.String getOUT_OF_BOUNDS_EXCEPTION_TEXT() {
1298     return OUT_OF_BOUNDS_EXCEPTION_TEXT;
      }
1300  /**
       * Insert the method's description here.
```

```
 * Creation date: (03-08-00 01:58:01)
 * @return java.lang.String
 */
public final static java.lang.String getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() {
   return PANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 20:20:00)
 * @return java.lang.String
 */
public final static java.lang.String getPLAIN_TEXT() {
   return PLAIN_TEXT;
}
/**
 * Insert the method's description here.
 * Creation date: (25-06-00 19:29:14)
 * @return java.lang.String
 */
public final static java.lang.String getPREFERENCES_FILE_NAME() {
   return PREFERENCES_FILE_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 19:40:40)
 * @return java.lang.String
 */
public final static java.lang.String getPROCESSING_DOCUMENTS() {
   return PROCESSING_DOCUMENTS;
}
/**
 * Insert the method's description here.
 * Creation date: (26-06-00 08:13:08)
 * @return java.lang.String
 */
public final static java.lang.String getPROPERTIES_HEADER() {
   return PROPERTIES_HEADER;
}
/**
 * Insert the method's description here.
 * Creation date: (21-08-00 15:49:38)
 * @return java.lang.String
 */
public final static java.lang.String getRAW_BITMAP() {
   return RAW_BITMAP;
}
/**
 * Insert the method's description here.
 * Creation date: (19-08-00 21:35:31)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */
public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName, String senderName)
{
   return senderName + "␣has␣requested␣that␣" + documentName + "␣be␣removed␣from␣the␣panorama.";
}
/**
 * Insert the method's description here.
 * Creation date: (05-09-00 15:15:12)
 * @param authorName java.lang.String
 * @param numDoks int
 */
public final static String getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST(String authorName, int numDoks)
{
   return "Removing␣" + numDoks + "␣documents␣from␣the␣panorama␣by␣request␣from␣" + authorName;
}
```

```
/**
 * Insert the method's description here.
 * Creation date: (19-08-00 21:32:47)
 * @return java.lang.String
 */
public final static java.lang.String getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() {
   return REMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 09:53:56)
 * @return java.lang.String
 */
public final static java.lang.String getREMOVING_DOCUMENTS() {
   return REMOVING_DOCUMENTS;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 00:35:49)
 * @return float
 */
public final static float getRENDER_CUTOFF() {
   return RENDER_CUTOFF;
}
/**
 * Insert the method's description here.
 * Creation date: (08-07-00 20:37:18)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_DOCUMENTS() {
   return REQUEST_DOCUMENTS;
}
/**
 * Insert the method's description here.
 * Creation date: (08-07-00 20:24:59)
 * @param numberOfDocuments String
 */
public static String getREQUESTING_DOCUMENTS_MESSAGE(int numberOfDocuments)
{
   return "Receiving␣and␣adding␣a␣total␣of␣" + String.valueOf( numberOfDocuments )+ "␣documents␣from␣the
        ␣other␣members␣of␣the␣group.";
}
/**
 * Insert the method's description here.
 * Creation date: (10-08-00 15:46:42)
 * @return java.lang.String
 */
public final static java.lang.String getREQUESTING_GROUPS_DIRECTORY() {
   return REQUESTING_GROUPS_DIRECTORY;
}
/**
 * Insert the method's description here.
 * Creation date: (08-08-00 09:52:01)
 * @return int
 */
public final static int getRESET_MESSAGE_AREA_DELAY() {
   return RESET_MESSAGE_AREA_DELAY;
}
/**
 * Insert the method's description here.
 * Creation date: (28-07-00 09:40:21)
 * @return java.awt.Color
 */
public final static java.awt.Color getRESIZE_RECTANGLE_PEN_COLOUR() {
   return RESIZE_RECTANGLE_PEN_COLOUR;
}
```

```
/**
 * Insert the method's description here.
 * Creation date: (19-06-00 22:01:04)
 * @return java.lang.String
 */
public final static java.lang.String getRETRIEVING_DOCUMENTS() {
   return RETRIEVING_DOCUMENTS;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 20:21:32)
 * @return java.lang.String
 */
public final static java.lang.String getRTF() {
   return RTF;
}
/**
 * Insert the method's description here.
 * Creation date: (14-07-00 14:38:06)
 * @return double
 */
public final static double getSCALE_FACTOR_SCALING() {
   return SCALE_FACTOR_SCALING;
}
/**
 * Insert the method's description here.
 * Creation date: (30-08-00 16:45:24)
 * @return java.lang.String
 */
public final static java.lang.String getSENDER_NAME() {
   return SENDER_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (30-08-00 17:24:57)
 * @return java.lang.String
 */
public final static java.lang.String getSHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX() {
   return SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX;
}
/**
 * Insert the method's description here.
 * Creation date: (25-06-00 16:35:18)
 * @return java.lang.String
 */
public final static java.lang.String getSIGNATURE() {
   return SIGNATURE;
}
/**
 * Insert the method's description here.
 * Creation date: (05-10-00 23:11:42)
 * @return int
 */
public final static int getTEXT_DOCUMENT_THRESHOLD() {
   return TEXT_DOCUMENT_THRESHOLD;
}
/**
 * Insert the method's description here.
 * Creation date: (11-08-00 00:22:49)
 * @return java.lang.String
 */
public final static java.lang.String getTITLE_REQUIRED_MESSAGE() {
   return TITLE_REQUIRED_MESSAGE;
}
/**
 * Insert the method's description here.
```

```
     * Creation date: (30-08-00 17:26:50)
1500  * @return java.lang.String
     */
1502 public final static java.lang.String getTRUE() {
        return TRUE;
1504 }
     /**
1506  * Insert the method's description here.
     * Creation date: (22-06-00 19:30:10)
1508  * @return int
     */
1510 public final static int getTYPE_INDEX() {
        return TYPE_INDEX;
1512 }
     /**
1514  * Insert the method's description here.
     * Creation date: (08-07-00 20:59:09)
1516  * @return java.lang.String
     * @param numberOfDocuments int
1518  * @param recipientName java.lang.String
     */
1520 public static String getUNICASTING_DOCUMENTS_MESSAGE(int numberOfDocuments, String recipientName) {
        return "New␣group␣member:␣unicasting␣" + String.valueOf( numberOfDocuments) + "␣documents␣to␣" +
             recipientName + "␣by␣request.";
1522 }
     /**
1524  * Insert the method's description here.
     * Creation date: (25-06-00 16:33:08)
1526  * @return java.lang.String
     */
1528 public final static java.lang.String getUPDATE_INTERVAL() {
        return UPDATE_INTERVAL;
1530 }
     /**
1532  * Insert the method's description here.
     * Creation date: (19-06-00 08:57:29)
1534  * @return int
     */
1536 public final static int getUPDATE_INTERVAL_IN_MILLISECS() {
        return UPDATE_INTERVAL_IN_MILLISECS;
1538 }
     /**
1540  * Insert the method's description here.
     * Creation date: (22-08-00 20:16:10)
1542  * @return int
     */
1544 public final static int getUPDATE_NOTIFICATION_THRESHOLD() {
        return UPDATE_NOTIFICATION_THRESHOLD;
1546 }
     /**
1548  * Insert the method's description here.
     * Creation date: (30-08-00 22:55:10)
1550  * @return java.lang.String
     */
1552 public final static java.lang.String getUPDATE_PANORAMA() {
        return UPDATE_PANORAMA;
1554 }
     /**
1556  * Insert the method's description here.
     * Creation date: (31-08-00 02:16:31)
1558  * @return java.lang.String
     */
1560 public final static java.lang.String getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() {
        return UPDATE_PANORAMA_BY_REQUEST_MESSAGE;
1562 }
     /**
```

```
1564  * Insert the method's description here.
      * Creation date: (21-08-00 23:11:12)
1566  * @return java.lang.String
      * @param documentName java.lang.String
1568  * @param senderName java.lang.String
      */
1570 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName) {
        return "Received␣update␣of␣" + documentName + ".␣Now␣updating...";
1572 }
     /**
1574  * Insert the method's description here.
      * Creation date: (21-08-00 23:11:12)
1576  * @return java.lang.String
      * @param documentName java.lang.String
1578  * @param senderName java.lang.String
      */
1580 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName, String senderName) {
        return "Received␣update␣of␣" + documentName + "␣from␣" + senderName + ".␣Now␣updating...";
1582 }
     /**
1584  * Insert the method's description here.
      * Creation date: (21-08-00 20:42:58)
1586  * @return java.lang.String
      */
1588 public final static java.lang.String getUPDATING_DOCUMENTS() {
        return UPDATING_DOCUMENTS;
1590 }
     /**
1592  * Insert the method's description here.
      * Creation date: (30-08-00 10:43:08)
1594  * @return java.lang.String
      */
1596 public final static java.lang.String getVISIBLE_DOCUMENTS_BOX_CAPTION() {
        return VISIBLE_DOCUMENTS_BOX_CAPTION;
1598 }
     /**
1600  * Insert the method's description here.
      * Creation date: (30-08-00 14:17:20)
1602  * @return int
      */
1604 public final static int getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() {
        return VISIBLE_DOCUMENTS_ICON_TEXT_GAP;
1606 }
     /**
1608  * Insert the method's description here.
      * Creation date: (01-09-00 07:50:17)
1610  * @return java.awt.Dimension
      */
1612 public final static java.awt.Dimension getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() {
        return VISIBLE_DOCUMENTS_ITEM_MAX_SIZE;
1614 }
     /**
1616  * Insert the method's description here.
      * Creation date: (31-08-00 00:15:59)
1618  * @return java.awt.Color
      */
1620 public final static java.awt.Color getVISIBLE_DOCUMENTS_SELECTION_COLOUR() {
        return VISIBLE_DOCUMENTS_SELECTION_COLOUR;
1622 }
     /**
1624  * Insert the method's description here.
      * Creation date: (30-08-00 14:26:51)
1626  * @return java.lang.String
      */
1628 public final static java.lang.String getVISIBLE_DOCUMENTS_TAB_SIZE() {
        return VISIBLE_DOCUMENTS_TAB_SIZE;
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (15-08-00 17:49:18)
 * @return java.lang.String
 */
public final static java.lang.String getZOOM_TO_OVERVIEW() {
   return ZOOM_TO_OVERVIEW;
}
/**
 * Insert the method's description here.
 * Creation date: (15-08-00 17:40:30)
 * @return java.lang.String
 */
public final static java.lang.String getZOOM_TO_OVERVIEW_ON_UPDATE() {
   return ZOOM_TO_OVERVIEW_ON_UPDATE;
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 19:10:52)
 * @return java.awt.Color
 */
public final static java.awt.Color getZOOM_VECTOR_COLOUR() {
   return ZOOM_VECTOR_COLOUR;
}
/**
 * Insert the method's description here.
 * Creation date: (31-07-00 18:02:18)
 * @return int
 */
public final static int getZOOM_VECTOR_WIDTH() {
   return ZOOM_VECTOR_WIDTH;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 19:51:35)
 */
public void initialize()
{

}
/**
 * Insert the method's description here.
 * Creation date: (23-02-01 08:42:21)
 * @param newFILE_TYPE_SYNONYMS java.util.Hashtable
 */
public static void setFILE_TYPE_SYNONYMS(java.util.Hashtable newFILE_TYPE_SYNONYMS) {
   FILE_TYPE_SYNONYMS = newFILE_TYPE_SYNONYMS;
}
}
```

Listing C.7: ClientPanorama.java

```
package clientapp;

/**
 * This class represents the client panorama
 * Creation date: (15-06-00 00:26:38)
 * @author:
 */
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.*;
import java.lang.reflect.InvocationTargetException;
import java.util.Hashtable;
import java.io.*;
import java.util.Enumeration;
```

```
import java.util.ArrayList;

import javax.swing.border.*;
import javax.swing.*;
import javax.swing.JProgressBar.*;
import javax.swing.Timer.*;
import javax.swing.SwingUtilities.*;

import edu.umd.cs.jazz.*;
import edu.umd.cs.jazz.util.*;
import edu.umd.cs.jazz.event.*;
import edu.umd.cs.jazz.component.*;
import java.awt.geom.*;

import java.awt.Point;
import javax.swing.event.MenuDragMouseListener;
import javax.swing.event.MenuDragMouseEvent;
import java.util.Collection;
import java.util.Iterator;

/** This class implements the panorama that occupies the center portion of the client application window
*/
public class ClientPanorama extends ZCanvas
{
   /** Event handler to ensure panorama updates when documents are added or removed
   */
   public class UpdateHandler extends ZGroupAdapter
   {
      public void nodeAdded( ZGroupEvent zge )
      {
         if ( zge.getChild() instanceof ZVisualLeaf )
         {
            validateLayout();
         }
      }
      public void nodeRemoved( ZGroupEvent zge )
      {
         if ( zge.getChild() instanceof ZVisualLeaf )
         {
            validateLayout();
         }
      }
   };

   /** Event handler triggered by a timer to ensure continous updating of the local documents at user
         determined intervals
   */
   public class UpdateActionListener implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         if ( isPanoramaLock() == false )
         {
            getUpdateTimer().stop();
            updateDocuments();
            getUpdateTimer().restart();
         }
      }
   }

   public class DocumentMouseListener extends ZMouseAdapter
   {
      public void mouseClicked( ZMouseEvent zme )
      {
         zme.consume();
      }
```

```
80      };

82
        public class CameraListener implements ZCameraListener
84      {
           public void viewChanged( ZCameraEvent zce )
86         {
           }
88      }

90      /** This listener handles user clicks on documents in the panorama.
        */
92      public class CameraMouseListener extends ZMouseAdapter
        {
94         public void mouseClicked ( ZMouseEvent zme )
           {
96            // if double click
              if ( zme.getClickCount() == 2 )
98            {
                 owner.getClientManager().clearTopicTree();
100              zoomToOverview();
              }
102           if ( isResizingNode() )
              {
104              doResize( zme );
              }
106           if ( isMovingNode() )
              {
108           // previousMousePosition = null;
                 setMovingNode( false );
110           }
              zme.consume();
112           popDownPopUpMenu();
           }
114     }

116     /** This listener handles clicks on the panorama canvas, i.e. outside any documents.
        */
118     public class PanoramaMouseListener extends ZMouseAdapter
        {
120        public void mouseClicked( ZMouseEvent zme )
           {
122           // if double click
              if ( zme.getClickCount() == 2 )
124           {
                 centerDocument( (ZVisualLeaf) zme.getNode() );
126              zme.consume();
              }
128        }

130        public void mousePressed( ZMouseEvent zme )
           {
132           if ( zme.getSource() instanceof ZVisualLeaf )
              {
134              // System.out.println(" Fyr ");
                 ( (ZSwing) ( (ZVisualLeaf) zme.getSource() ).getVisualComponent() ).getComponent().
                     dispatchEvent( zme );
136           }

138           if ( isResizingNode() )
              {
140              doResize( zme );
              }
142           if ( isMovingNode() )
              {
144              // lock node to current position if auto-lock is enabled
```

```
            getCurrentlyClickedNode().putClientProperty( ClientConstants.getNODE_LOCKED(), ClientInfo.
                getPreferences().getProperty( ClientConstants.getAUTOLOCK_NODES() ));
146         // previousMousePosition = null;
            setMovingNode( false );
148         }
            // if right click, bring up the document pop-up menu
150         if ( zme.isMetaDown() )
            {
152           popUpPopUpMenu( (ZVisualLeaf) zme.getNode(), zme.getPoint() );
              zme.consume();
154         }
            else
156         {
              popDownPopUpMenu();
158         }
          }
160
          public void mouseReleased( ZMouseEvent zme )
162       {
          }
164     }

166     public class PanoramaMouseMotionListener extends ZMouseMotionAdapter
        {
168       public void mouseMoved( ZMouseEvent zme )
          {  // a cleaner solution would be to use the mouseExited event in a mouselistener on the popup menu
               , but that
170         // seemed to fail due to improper generation of motion events
            // popDownPopUpMenu();
172         if ( isResizingNode() == true )
            {  ZBounds bounds = currentlyClickedNode.getVisualComponentBounds();
174           Point mousePosition = new Point( zme.getX(), zme.getY() );
              Point upperLeftCorner = new Point( (int)bounds.getMinX(), (int)bounds.getMinY() );
176           // convert to global coordinates
              getCamera().cameraToLocal( mousePosition, currentlyClickedNode );
178           //getCamera().cameraToLocal( upperLeftCorner, null );
              Dimension newSize = new Dimension();
180           newSize.setSize(
                (int) mousePosition.getX() - (int) upperLeftCorner.getX(),
182             (int) mousePosition.getY() - (int) upperLeftCorner.getY() );
              ( (PZSwing)currentlyClickedNode.getVisualComponent() ).getComponent().setPreferredSize(
                  newSize );
184           ( (PZSwing)currentlyClickedNode.getVisualComponent() ).computeBounds();
              currentlyClickedNode.reshape();
186           revalidate();
              getDrawingSurface().repaint();
188         }
            if ( isMovingNode() == true )
190         {
              // I tried a variety of approaches to implementing the below translation code for panorama
                  nodes, i.e. documents.
192           // These included calculating the difference between the position between the most current
                  mouse click and the preceding one,
              // and then using that as the translation measure for the node in question. That scheme
                  failed, however, and so did my next
194           // attempt which was to convert the mouse positions into the local coordinate system for the
                  clicked node and proceed as before.
              // This failed irrespective of whether I used the globalToLocal methods of the node or the
                  cameraToLocal methods of the camera, which
196           // is hardly surprising given that the latter uses calls to the former to fulfill its purpose
                  . Finally, I tried using the inverse transform
              // of the camera directly, i.e. without intermediary function calls and that did the trick !
198             Point globalPoint = new Point();
                getCamera().getInverseViewTransform().transform( zme.getPoint(), globalPoint );
200             float x = (float)globalPoint.getX();
                float y = (float)globalPoint.getY();
```

```
202                currentlyClickedNode.editor().getTransformGroup().setTranslation( x, y );
           }
204
        }
206     }

208     /** Handler for the pop-up menu activated by right clicking documents in the panorama
        */
210     public class PopUpActionListener implements ActionListener
        {
212        public void actionPerformed( ActionEvent ae )
           {
214           JMenuItem menuItem = null;
              // the below test makes explicit what is implicitly assumed: that the originator of the event
                  was a popup menu item
216           if ( ae.getSource() instanceof JMenuItem )
              {
218              menuItem = (JMenuItem) ae.getSource();
              }
220           else
              {
222              return;
              }
224           getDocumentMenu().setVisible( false );
              if ( menuItem == getResizeItem() )
226           {
                 // System.out.println( "resize" );
228              resizeDocument();
              }
230           else if ( menuItem == getMoveItem() )
              {
232              // System.out.println( "move" );
                 moveDocument();
234           }
           /* if ( menuItem == getZoom() )
236           {
                 // System.out.println( "zoom" );
238              zoomDocument();
              }
240        */ else if ( menuItem == getCopyTextItem() )
              {
242              // System.out.println( "copy text" );
                 owner.getClientManager().copyText( currentlyClickedNode );
244           }
              else if ( menuItem == getIvjLockItem() )
246           {
                 // System.out.println( "lock item" );
248              lockNode( currentlyClickedNode );
              }
250           else if ( menuItem == getIvjUnlockAllItem() )
              {
252              // System.out.println(" unlockall item" );
                 unLockAll();
254           }
              else if ( menuItem == getIvjAutoLockAllItem() )
256           {
                 // System.out.println(" autolockall item" );
258              autoLock();
              }
260           else if ( menuItem == getIvjArrangeAllItem() )
              {
262              // System.out.println(" arrangeall item" );
                 validateLayout();
264              zoomToOverview();
              }
266        }
```

```
    }

    public class PopUpMenuMouseHandler extends MouseAdapter
    {
       public void mouseExited( MouseEvent me )
       {
          // The below was commented out in favour of a MouseMotionListener.mouseMoved event handler (
               below)
          // The reason was that mouseExited events didn't seem to be generated when the mouse cursor was
               moved quickly off
          // the source component
          // getDocumentMenu().setVisible( false );
       }
    }

    PanoramaMouseListener panoramaMouseListener = new PanoramaMouseListener();
    PanoramaMouseMotionListener panoramaMouseMotionListener = new PanoramaMouseMotionListener();
    float initialScale = 0;
    ZVisualLeaf newLeaf;
    ZPolyline zoomVector;
    static Point previousMousePosition = null;
    private ClientApplication owner;
    private java.util.Hashtable documents = new Hashtable();
    private java.util.Hashtable localDocuments = new Hashtable();
    // value modified in popUpPopUpMenu and popDownPopUpMenu
 private edu.umd.cs.jazz.ZVisualLeaf currentlyClickedNode = null;
    private boolean movingNode = false;
    private boolean zoomingNode = false;
    private edu.umd.cs.jazz.util.ZBounds boundsBeforeResizing = null;
    private boolean resizingNode = false;
    private JSeparator ivjJSeparator2 = null;
    private JMenuItem ivjMoveItem = null;
    private JMenuItem ivjResizeItem = null;
    private JPopupMenu ivjDocumentMenu = null;
    private JMenuItem ivjCopyTextItem = null;
    private javax.swing.Timer updateTimer = null;
    private ClientPanoramaLayoutManager layoutManager = null;
    private boolean panoramaLock = false;
    private javax.swing.JMenuItem ivjUnlockAllItem = null;
    private javax.swing.JCheckBoxMenuItem ivjAutoLockAllItem = null;
    private javax.swing.JSeparator ivjSeparator3 = null;
    private javax.swing.JSeparator ivjSeparator4 = null;
    private javax.swing.JMenuItem ivjArrangeAllItem = null;
    private javax.swing.JCheckBoxMenuItem ivjLockItem = null;
 /**
  * ClientPanorama constructor comment.
  */
 public ClientPanorama() {
    super();
    initialize();
 }
 /**
  * ClientPanorama constructor comment.
  * @param aRoot edu.umd.cs.jazz.ZRoot
  * @param layer edu.umd.cs.jazz.ZLayerGroup
  */
 public ClientPanorama(edu.umd.cs.jazz.ZRoot aRoot, edu.umd.cs.jazz.ZLayerGroup layer) {
    super(aRoot, layer);
 }
 /**
  * Add a new document to the panorama by inserting a leaf node and decorating it with the properties of
       the document, including its contents (passed as a byte arry).
  * Creation date: (13-08-00 01:01:23)
  * @return edu.umd.cs.jazz.ZVisualLeaf
  * @param rawFileContents byte[]
  * @param senderName java.lang.String
```

```
330  * @param authorName java.lang.String
     * @param documentName java.lang.String
332  */
     private ZVisualLeaf addDocument(byte[] rawFileContents, String senderName, String authorName, String
         documentNameAndPath, String documentName)
334  {
       ZVisualLeaf leaf;
336    ZTransformGroup transform;
       PZSwing swing;
338    JScrollPane pane;

340    //System.out.println( "Document name " + documentName + "\n");
       //System.out.println( "Document name and path " + documentNameAndPath + "\n");
342    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
       String extension = fqPath.substring( fqPath.lastIndexOf('.')).toLowerCase();
344    String mediaType = (String)ClientConstants.getMEDIA_TYPES().get(extension);
       if ( mediaType == null )
346    {
         mediaType = ClientConstants.getALL_FILES();
348    }
       leaf = new ZVisualLeaf();
350    leaf.putClientProperty( ClientConstants.getAUTHOR_NAME(), authorName );
       leaf.putClientProperty( ClientConstants.getDOCUMENT_NAME(), documentNameAndPath );
352    leaf.putClientProperty( ClientConstants.getSENDER_NAME(), senderName );
       if (mediaType.equalsIgnoreCase( ClientConstants.getPLAIN_TEXT() ))
354    {  JTextArea textArea = new JTextArea( new String( rawFileContents ));
         textArea.setEditable( false );
356      pane = new JScrollPane( textArea );
       }
358    else if (mediaType.equalsIgnoreCase( ClientConstants.getHTML() ))
       {  JEditorPane editorPane = new JEditorPane( ClientConstants.getHTML(), new String( rawFileContents )
           );
360      editorPane.setEditable( false );
         pane = new JScrollPane( editorPane );
362    }
       else if (mediaType.equalsIgnoreCase( ClientConstants.getRTF() ))
364    {  JEditorPane editorPane = new JEditorPane( ClientConstants.getRTF(), new String( rawFileContents ))
           ;
         editorPane.setEditable( false );
366      pane = new JScrollPane( editorPane );
       }
368    else if (mediaType.equalsIgnoreCase( ClientConstants.getBITMAP() ))
       {  // the unformatted byte array containing the image data is stored separately since extracting it
           from the
370      // ImageIcon after construction appears to be impossible - may be changed if I figure out a way
             that doesn't
         // incur a hefty performance penalty
372      pane = new JScrollPane( new JLabel( new ImageIcon(rawFileContents) ));
         pane.putClientProperty( ClientConstants.getRAW_BITMAP(), rawFileContents );
374    }
       // default case
376    else
       {
378      if ( rawFileContents.length < ClientConstants.getTEXT_DOCUMENT_THRESHOLD() )
         {
380        JTextArea textArea = new JTextArea( new String( rawFileContents ));
           textArea.setEditable( false );
382        pane = new JScrollPane( textArea );
         }
384      else
         {
386        pane = new JScrollPane( new JLabel( new ImageIcon(rawFileContents) ));
         }
388
       }
390    // add informative label to top of scroll pane, complete with bevelled border
```

```
392     JTextArea newLabel = newDocumentLabel( ClientConstants.getDOCUMENT_LABEL( documentNameAndPath,
            authorName, rawFileContents.length, new java.util.Date() ));
        pane.setColumnHeaderView( newLabel );
394     pane.setPreferredSize( getLayoutManager().getDocumentSize() );
        pane.setMaximumSize( pane.getPreferredSize() );
396     swing = new PZSwing(this, pane);
        pane.scrollRectToVisible( pane.getViewportBorderBounds() );
398     leaf.setVisualComponent( swing );
        leaf.putClientProperty( ClientConstants.getDOCUMENT_PATH(), fqPath );
400     // KAN OPTIMERES: det er ikke øndvendigt med en instans til hver node - øprv evt. med en enkelt, delt
            instans til alle nodes
        // catch mouse events pertaining to this node and process them appropriately by adding a custom mouse
            listener
402     leaf.addMouseListener( panoramaMouseListener );
        leaf.addMouseMotionListener( panoramaMouseMotionListener );
404     owner.addDocumentToVisibleList( senderName, authorName, documentNameAndPath, documentName );
        getLayer().addChild(leaf);
406     return leaf;
    }
408 /**
     * Add document originating from other group members than the one using the client making the call to
         this function.
410  * Creation date: (13-08-00 00:58:00)
     * @param rawFileContents byte[]
412  * @param senderName java.lang.String
     * @param authorName java.lang.String
414  * @param documentName java.lang.String
     */
416 public ZVisualLeaf addGlobalDocument( byte[] rawFileContents, String senderName, String authorName,
         String documentNameAndPath, String documentName)
    {
418     ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath,
            documentName );
        String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
420     getDocuments().put( fqPath, newLeaf );
        return newLeaf;
422 }
    /**
424  * Add document added by the user of the client calling this method.
     * Creation date: (13-08-00 01:05:09)
426  * @param rawFileContents byte[]
     * @param senderName java.lang.String
428  * @param authorName java.lang.String
     * @param documentName java.lang.String
430  */
    public ZVisualLeaf addLocalDocument(byte[] rawFileContents, String senderName, String authorName, String
          documentNameAndPath, long lastModified)
432 {
        String documentName = documentNameAndPath.substring( documentNameAndPath.lastIndexOf( File.separator
            )+ 1 );
434     ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath,
            documentName );
        newLeaf.putClientProperty( ClientConstants.getLAST_MODIFIED(), new Long( lastModified ));
436     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
        getLocalDocuments().put( fqPath, newLeaf );
438     return newLeaf;
    }
440 /**
     * Insert the method's description here.
442  * Creation date: (14-10-00 10:39:16)
     */
444 public void autoLock()
    {
446     if ( getIvjAutoLockAllItem().getState() == true )
        {
```

```
448         ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.
                getTRUE() );
      }
450   else
      {
452         ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.
                getFALSE() );
      }
454   }
      /**
456    * Begin a zoom and pan sequence to have the node passed as argument centered and request the topic tree
           corresponding
       * to that node from the server.
458    * Creation date: (07-08-00 09:19:29)
       * @param zme edu.umd.cs.jazz.event.ZMouseEvent
460    */
      public void centerDocument( ZVisualLeaf clickedNode )
462   {
         String path = (String) clickedNode.getClientProperty( ClientConstants.getDOCUMENT_PATH() );
464      if ( ( path.equals( owner.getClientManager().getClientInfo().getCurrentDocumentName() )== false ))
         {
466         owner.getClientManager().requestTopicTree( path );
             //System.out.println( "Clicked path: " + path );
468      }
         zoomAndPanToNode( clickedNode );
470   }
      /**
472    * Convenience method to allow call by documentID rather than node
       * Creation date: (30-08-00 23:51:07)
474    * @param documentID java.lang.String
       */
476   public void centerDocument(String documentID)
      {
478      if ( getLocalDocuments().containsKey( documentID ))
         {
480         centerDocument( (ZVisualLeaf) getLocalDocuments().get( documentID ));
         }
482      else
         {
484         centerDocument( (ZVisualLeaf) getDocuments().get( documentID ));
         }
486
      }
488   /**
       * Compute the bounding box for the set of nodes passed as argument and return it as a rectangle object.
490    * I first tried using the getGlobalBounds for each node but that seemed to produce a wrongly
       * translated, although correctly sized, bounding box. The below use of getBounds followed by a
           localToGlobal transform seems to produce
492    * a correct result. Maybe the effect is due to value caching or spurious translations by function
           getGlobalBounds()...
       * Creation date: (12-10-00 23:59:36)
494    * @return java.awt.Rectangle
       * @param nodes edu.umd.cs.jazz.ZNode[]
496    */
      public java.awt.Rectangle computeOverviewRectangle(ZNode[] nodes)
498   {
      /* float minX = Float.MAX_VALUE, minY = Float.MAX_VALUE, maxX = Float.MIN_VALUE, maxY = Float.MIN_VALUE;
500      float xcandidate, ycandidate;
         for ( int i = 0 ; i < nodes.length; i++ )
502      {
            ZBounds bounds = nodes[i].getBounds();
504         nodes[i].localToGlobal( bounds );
            xcandidate = (int)bounds.getMinX();
506         ycandidate = (int)bounds.getMinY();
            minX = Math.abs( xcandidate )< Math.abs( minX )? xcandidate : minX;
508         minY = Math.abs( ycandidate )< Math.abs( minY )? ycandidate : minY;
```

```
      xcandidate = (int)bounds.getMaxX();
      ycandidate = (int)bounds.getMaxY();
      maxX = Math.abs( xcandidate )> Math.abs( maxX )? xcandidate : maxX;
      maxY = Math.abs( ycandidate )> Math.abs( maxY )? ycandidate : maxY;
   }
   java.awt.Rectangle overviewRectangle = new java.awt.Rectangle( (int)minX, (int)minY, (int)maxX, (int)
        maxY );
   return overviewRectangle;
*/
   return (java.awt.Rectangle) getLayer().getGlobalBounds().getBounds();
}
/**
 * Remove the panorama pop-up menu.
 * Creation date: (20-07-00 17:59:56)
 */
public void dismissPopUpMenu()
{
   getDocumentMenu().setVisible( false );
}
/**
 * Perform a resize operation.
 * Creation date: (31-07-00 20:02:40)
 * @param zMouseEvent edu.umd.cs.jazz.event.ZMouseEvent
 */
public void doResize(ZMouseEvent zme)
{
   // if right click, return to original size
   if ( zme.isMetaDown() == true )
   {  // use bounds of resize rectangle to reshape active scenegraph node and notify relevant objects
        that this has been done
      Dimension newSize = new Dimension();
      //getCamera().cameraToLocal( bounds, currentlyClickedNode );
      newSize.setSize(
         (int) boundsBeforeResizing.getMaxX() - (int) boundsBeforeResizing.getMinX(),
         (int) boundsBeforeResizing.getMaxY() - (int) boundsBeforeResizing.getMinY() );
      ( (PZSwing)currentlyClickedNode.getVisualComponent() ).getComponent().setPreferredSize( newSize );
      ( (PZSwing)currentlyClickedNode.getVisualComponent() ).computeBounds();
      currentlyClickedNode.reshape();
   }
   revalidate();
   getDrawingSurface().repaint();
   setResizingNode( false );
}
/**
 * Carry out a zooming operation
 * Creation date: (31-07-00 21:05:01)
 */
public void doZooming( ZMouseEvent zme )
{
   if ( zme.isMetaDown() == true )
   {
      currentlyClickedNode.editor().getTransformGroup().setScale( initialScale );
   }
   previousMousePosition = null;
   setZoomingNode( false );
   ( (PZSwing) currentlyClickedNode.getVisualComponent() ).setZoomingNode( null );
// getLayer().removeChild( newLeaf );
   revalidate();
   getDrawingSurface().repaint();
}
/**
 * Insert all panorama documents, both local and global into an array and return it.
 * Creation date: (14-10-00 16:42:03)
 * @return edu.umd.cs.jazz.ZNode[]
 */
public ZNode[] getAllDocumentsAsArray()
```

```
{
   ZNode nodes[] = new ZNode[ getLocalDocuments().size() + getDocuments().size() ];
   int i = 0;
   for (int j = 0 ; j < getLocalDocuments().size(); j++, i++ )
   {
      nodes[ i ] = (ZNode) getLocalDocuments().values().toArray()[ j ];
   }
   for ( int j = 0 ; j < getDocuments().size(); j++, i++ )
   {
      nodes[ i ] = (ZNode) getDocuments().values().toArray()[ j ];
   }
   return nodes;
}
/**
 * Insert the method's description here.
 * Creation date: (31-07-00 19:43:21)
 * @return edu.umd.cs.jazz.util.ZBounds
 */
public edu.umd.cs.jazz.util.ZBounds getBoundsBeforeResizing() {
   return boundsBeforeResizing;
}
/**
 * Return the CopyText property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getCopyTextItem() {
   if (ivjCopyTextItem == null) {
      try {
         ivjCopyTextItem = new javax.swing.JMenuItem();
         ivjCopyTextItem.setName("CopyTextItem");
         ivjCopyTextItem.setMnemonic('C');
         ivjCopyTextItem.setText("Copy␣text");
         // user code begin {1}
         ivjCopyTextItem.addActionListener( new PopUpActionListener() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjCopyTextItem;
}
/**
 * Insert the method's description here.
 * Creation date: (28-07-00 00:50:07)
 * @return edu.umd.cs.jazz.ZVisualLeaf
 */
public edu.umd.cs.jazz.ZVisualLeaf getCurrentlyClickedNode() {
   return currentlyClickedNode;
}
/**
 * Helper method to return a document node, irrespective of whether it is global or local
 * Creation date: (14-10-00 11:47:11)
 * @return edu.umd.cs.jazz.ZVisualLeaf
 * @param clientName java.lang.String
 * @param documentName java.lang.String
 */
public ZVisualLeaf getDocument(String clientName, String documentName)
{
   String fqName = ClientConstants.createFullyQualifiedDocumentName( clientName, documentName );
   ZVisualLeaf returnLeaf = null;
   if ( localDocuments.get( fqName )!= null )
   {
      returnLeaf = (ZVisualLeaf) getLocalDocuments().get( fqName );
```

```
      }
      else
      {
         returnLeaf = (ZVisualLeaf) getDocuments().get( fqName );
      }
      return returnLeaf;
   }
   /**
    * Return the PopupMenu property value.
    * @return javax.swing.JPopupMenu
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private JPopupMenu getDocumentMenu() {
      if (ivjDocumentMenu == null) {
         try {
            ivjDocumentMenu = new JPopupMenu();
            ivjDocumentMenu.setName("DocumentMenu");
            ivjDocumentMenu.add(getResizeItem());
            ivjDocumentMenu.add(getMoveItem());
            ivjDocumentMenu.add(getIvjLockItem());
            ivjDocumentMenu.add(getJSeparator2());
            ivjDocumentMenu.add(getCopyTextItem());
            ivjDocumentMenu.add(getIvjSeparator3());
            ivjDocumentMenu.add(getIvjUnlockAllItem());
            ivjDocumentMenu.add(getIvjArrangeAllItem());
            ivjDocumentMenu.add(getIvjAutoLockAllItem());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjDocumentMenu;
   }
   /**
    * Insert the method's description here.
    * Creation date: (08-07-00 01:30:08)
    * @return java.util.Hashtable
    */
   public java.util.Hashtable getDocuments() {
      return documents;
   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 13:35:55)
    * @return javax.swing.JMenuItem
    */
   public javax.swing.JMenuItem getIvjArrangeAllItem()
   {
      if (ivjArrangeAllItem == null) {
         try {
            ivjArrangeAllItem = new javax.swing.JMenuItem();
            ivjArrangeAllItem.setName("ArrangeAllItem");
            ivjArrangeAllItem.setMnemonic('R');
            ivjArrangeAllItem.setText("Arrange␣all");
            // user code begin {1}
            ivjArrangeAllItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
```

```
      return ivjArrangeAllItem;

   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 09:53:08)
    * @return javax.swing.JCheckBoxMenuItem
    */
   public javax.swing.JCheckBoxMenuItem getIvjAutoLockAllItem() {
      if (ivjAutoLockAllItem == null) {
         try {
            ivjAutoLockAllItem = new javax.swing.JCheckBoxMenuItem();
            ivjAutoLockAllItem.setName("AutoLockAllItem");
            ivjAutoLockAllItem.setMnemonic('A');
            ivjAutoLockAllItem.setText("Auto-lock␣all");
            // user code begin {1}
            ivjAutoLockAllItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjAutoLockAllItem;
   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 13:42:49)
    * @return javax.swing.JCheckBoxMenuItem
    */
   public javax.swing.JCheckBoxMenuItem getIvjLockItem()
   {
      if (ivjLockItem == null) {
         try {
            ivjLockItem = new javax.swing.JCheckBoxMenuItem();
            ivjLockItem.setName("LockItem");
            ivjLockItem.setMnemonic('L');
            ivjLockItem.setText("Lock");
            // user code begin {1}
            ivjLockItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjLockItem;

   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 10:14:35)
    * @return javax.swing.JSeparator
    */
   public javax.swing.JSeparator getIvjSeparator3() {
      if (ivjSeparator3 == null) {
         try {
            ivjSeparator3 = new javax.swing.JSeparator();
            ivjSeparator3.setName("JSeparator3");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
```

```
            handleException(ivjExc);
         }
      }
      return ivjSeparator3;
   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 10:14:58)
    * @return javax.swing.JSeparator
    */
   public javax.swing.JSeparator getIvjSeparator4() {
      if (ivjSeparator4 == null) {
         try {
            ivjSeparator4 = new javax.swing.JSeparator();
            ivjSeparator4.setName("JSeparator4");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjSeparator4;

   }
   /**
    * Insert the method's description here.
    * Creation date: (14-10-00 09:51:30)
    * @return javax.swing.JMenuItem
    */
   public javax.swing.JMenuItem getIvjUnlockAllItem() {
      if (ivjUnlockAllItem == null) {
         try {
            ivjUnlockAllItem = new javax.swing.JMenuItem();
            ivjUnlockAllItem.setName("UnlockAllItem");
            ivjUnlockAllItem.setMnemonic('N');
            ivjUnlockAllItem.setText("Unlock␣all");
            // user code begin {1}
            ivjUnlockAllItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjUnlockAllItem;

   }
   /**
    * Return the JSeparator2 property value.
    * @return javax.swing.JSeparator
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JSeparator getJSeparator2() {
      if (ivjJSeparator2 == null) {
         try {
            ivjJSeparator2 = new javax.swing.JSeparator();
            ivjJSeparator2.setName("JSeparator2");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
```

```
         }
838     }
        return ivjJSeparator2;
840  }
     /**
842   * Insert the method's description here.
      * Creation date: (29-08-00 01:03:17)
844   * @return clientapp.ClientPanoramaLayout
      */
846  public ClientPanoramaLayoutManager getLayoutManager() {
        if( layoutManager == null )
848     {
           setLayoutManager( new GridPanoramaLayoutManager() );
850     }
        return layoutManager;
852  }
     /**
854   * Insert the method's description here.
      * Creation date: (26-07-00 11:38:26)
856   * @return java.util.Hashtable
      */
858  public java.util.Hashtable getLocalDocuments()
     {
860     return localDocuments;
     }
862  /**
      * Return the MoveItem property value.
864   * @return javax.swing.JMenuItem
      */
866  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JMenuItem getMoveItem() {
868     if (ivjMoveItem == null) {
           try {
870           ivjMoveItem = new javax.swing.JMenuItem();
              ivjMoveItem.setName("MoveItem");
872           ivjMoveItem.setMnemonic('M');
              ivjMoveItem.setText("Move");
874           // user code begin {1}
              ivjMoveItem.addActionListener( new PopUpActionListener() );
876           // user code end
           } catch (java.lang.Throwable ivjExc) {
878           // user code begin {2}
              // user code end
880           handleException(ivjExc);
           }
882     }
        return ivjMoveItem;
884  }
     /**
886   * Insert the method's description here.
      * Creation date: (18-06-00 17:41:38)
888   * @return clientapp.ClientApplication
      */
890  public ClientApplication getOwner() {
        return owner;
892  }
     /**
894   * Return the ResizeItem property value.
      * @return javax.swing.JMenuItem
896   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
898  private javax.swing.JMenuItem getResizeItem() {
        if (ivjResizeItem == null) {
900        try {
              ivjResizeItem = new javax.swing.JMenuItem();
902           ivjResizeItem.setName("ResizeItem");
```

```
            ivjResizeItem.setMnemonic('R');
904         ivjResizeItem.setText("Resize");
            // user code begin {1}
906         ivjResizeItem.addActionListener( new PopUpActionListener() );
            // user code end
908       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
910         // user code end
            handleException(ivjExc);
912       }
        }
914     return ivjResizeItem;
    }
916 /**
     * Insert the method's description here.
918  * Creation date: (22-08-00 15:44:15)
     * @return javax.swing.Timer
920  */
    public javax.swing.Timer getUpdateTimer()
922 {
        if ( updateTimer == null)
924     {  // start update process in separate thread - multiply by 1000 to get the preferred or default
             setting in milliseconds.
           updateTimer = new Timer( Integer.parseInt( ClientInfo.getPreferences().getProperty(
               ClientConstants.getUPDATE_INTERVAL() ))* 1000,
926                         new UpdateActionListener() );
        }
928     return updateTimer;
    }
930 /**
     * Called whenever the part throws an exception.
932  * @param exception java.lang.Throwable
     */
934 private void handleException(java.lang.Throwable exception) {

936     /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
938     // exception.printStackTrace(System.out);
    }
940 /**
     * Insert the method's description here.
942  * Creation date: (16-06-00 17:44:24)
     */
944 public void initialize()
    {
946     try {
           // user code begin {1}
948        // user code end
           setName("ClientPanorama");
950        setSize(1, 1);
        } catch (java.lang.Throwable ivjExc) {
952        handleException(ivjExc);
        }
954     // user code begin {2}
        getCameraNode().addMouseListener( new CameraMouseListener() );
956     getCameraNode().addMouseMotionListener( new PanoramaMouseMotionListener() );
        getCamera().addCameraListener( new CameraListener() );
958     zoomEventHandler = new PZoomHandler( getCameraNode() );
        zoomEventHandler.setActive( true );
960     panEventHandler = new PZPanEventHandler( getCameraNode() );
        panEventHandler.setActive( true );
962     swingEventHandler = new ZSwingEventHandler( this, getCameraNode() );
        swingEventHandler.setActive( true );
964     getDrawingSurface().setRenderQuality( Integer.parseInt( (String) ClientConstants.getDISPLAY_QUALITIES
             ().
```

```
                                 get( ClientInfo.getPreferences().getProperty( ClientConstants.
                                      getDISPLAY_QUALITY() ))));
966     getUpdateTimer().start();
        getLayoutManager().setSettings( ClientInfo.getPreferences() );
968     //getLayer().addGroupListener( new UpdateHandler() );
        // user code end
970  }
     /**
972   * Insert the method's description here.
      * Creation date: (29-07-00 18:22:23)
974   * @return boolean
      */
976  public boolean isMovingNode() {
        return movingNode;
978  }
     /**
980   * Insert the method's description here.
      * Creation date: (07-09-00 21:16:49)
982   * @return boolean
      */
984  public boolean isPanoramaLock() {
        return panoramaLock;
986  }
     /**
988   * Insert the method's description here.
      * Creation date: (31-07-00 20:04:03)
990   * @return boolean
      */
992  public boolean isResizingNode() {
        return resizingNode;
994  }
     /**
996   * Insert the method's description here.
      * Creation date: (29-07-00 19:19:43)
998   * @return boolean
      */
1000 public boolean isZoomingNode() {
        return zoomingNode;
1002 }
     /**
1004  * Insert the method's description here.
      * Creation date: (14-10-00 10:37:47)
1006  * @param node edu.umd.cs.jazz.ZVisualLeaf
      */
1008 public void lock(ZNode node)
     {
1010    node.putClientProperty( ClientConstants.getNODE_LOCKED(), ClientConstants.getTRUE() );
     }
1012 /**
      * Insert the method's description here.
1014  * Creation date: (14-10-00 13:47:46)
      * @param node edu.umd.cs.jazz.ZNode
1016  */
     public void lockNode(ZNode node)
1018 {
        if ( getIvjLockItem().getState() == true )
1020    {
           lock( node );
1022    }
        else
1024    {
           unlock( node );
1026    }
     }
1028 /**
      * main entrypoint - starts the part when it is run as an application
```

```
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      JFrame frame = new javax.swing.JFrame();
      ClientPanorama aClientPanorama;
      aClientPanorama = new ClientPanorama();
      frame.setContentPane(aClientPanorama);
      frame.setSize(aClientPanorama.getSize());
      frame.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      frame.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣clientapp.ClientPanorama");
      exception.printStackTrace(System.out);
   }
}
/**
 * Translate the document associated with the currently active node in the direction and by the amount
     indicated by the
 * user's mouse movements.
 * Creation date: (28-07-00 01:13:44)
 */
public void moveDocument()
{
   setMovingNode( true );
   previousMousePosition = null;
}
/**
 * Insert the method's description here.
 * Creation date: (22-08-00 11:18:54)
 * @return javax.swing.JTextArea
 * @param text java.lang.String
 */
public JTextArea newDocumentLabel(String text)
{
   JTextArea newLabel = new JTextArea( text );
   newLabel.setEditable( false );
   newLabel.setBackground( Color.orange );
// newLabel.setSize( new Dimension( Integer.parseInt( ClientConstants.getDEFAULT_DOCUMENT_WIDTH() ), (
     int) newLabel.getSize().getHeight() ));
   Border bevelBorder = BorderFactory.createRaisedBevelBorder();
   Border matteBorder = BorderFactory.createMatteBorder( ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                       ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                       ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                       ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                       newLabel.getBackground() );
   Border compoundBorder = BorderFactory.createCompoundBorder( bevelBorder, matteBorder );
   newLabel.setBorder( compoundBorder );
   newLabel.setLineWrap( true );
   return newLabel;
}
/**
 * Insert the method's description here.
 * Creation date: (28-07-00 01:02:00)
 */
public void popDownPopUpMenu()
{
   getDocumentMenu().setVisible( false );
// setCurrentlyClickedNode( null );
}
/**
```

```
 * Opens the document pop-up menu, modifying it to reflect the type of document on which the menu was
     invoked
 * Creation date: (20-07-00 16:48:11)
 */
public void popUpPopUpMenu( ZVisualLeaf node, Point position )
{
   setCurrentlyClickedNode( node );
   JScrollPane clickedPane = (JScrollPane) ( (PZSwing) node.getVisualComponent() ).getComponent();
   // enable text copying only if the clicked node displays text, either formatted or plain
   if ( clickedPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
   {
      getCopyTextItem().setEnabled( true );
   }
   else
   {
      getCopyTextItem().setEnabled( false );
   }
   boolean state = ClientInfo.getPreferences().getProperty( ClientConstants.getAUTOLOCK_NODES() ).equals
        ( ClientConstants.getTRUE() )? true : false;
   getIvjAutoLockAllItem().setState( state );
   if (getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() )!= null )
   {
      state = getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() ).equals(
           ClientConstants.getTRUE() )? true : false;
   }
   else
   {
      state = false;
   }
   getIvjLockItem().setState( state );
   getDocumentMenu().show( this, (int)position.getX(), (int)position.getY() );
   getDocumentMenu().setVisible( true );
}
/**
 * Remove document originating from other group member.
 * Creation date: (03-08-00 13:13:00)
 * @param path java.lang.String
 */
public void removeGlobalDocument(String senderName, String documentName )
{
   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
   ZNode topNode = ( (ZNode) getDocuments().get( fqPath )).editor().getTop();
   if ( topNode != null )
   {
      topNode.getParent().removeChild( topNode );
   }
   owner.removeDocumentFromVisibleList( senderName, documentName );
   getDocuments().remove( fqPath );
}
/**
 * Remove document inserted by the user the client making the call to this method.
 * Creation date: (23-06-00 08:39:46)
 * @param path java.lang.String
 */
public void removeLocalDocument(String senderName, String documentName )
{
   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
   ZNode topNode = ( (ZNode) getLocalDocuments().get( fqPath )).editor().getTop();
   if ( topNode != null )
   {
      topNode.getParent().removeChild( topNode );
      getLocalDocuments().remove( fqPath );
      // System.out.println( "Removed: " + fqPath );
   }
   owner.removeDocumentFromVisibleList( senderName, documentName );
}
```

```
1156  /**
       *
1158   * Creation date: (28-07-00 01:13:33)
       */
1160  public void resizeDocument()
      {
1162     boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
         setResizingNode( true );
1164  }
      /**
1166   * Insert the method's description here.
       * Creation date: (14-07-00 10:55:28)
1168   * @param scaleFactor float
       * @param visualLeaf edu.umd.cs.jazz.ZVisualLeaf
1170   */
      public void scaleDocument(double scaleFactor, ZVisualLeaf visualLeaf)
1172  {
         ZTransformGroup zTransformGroup = visualLeaf.editor().getTransformGroup();
1174     AffineTransform affineTransform = zTransformGroup.getTransform();

1176     // System.out.println( String.valueOf( scaleFactor ));
         // System.out.println( String.valueOf( getCamera().getScale() ));
1178
         //scaleFactor = zTransformGroup.getScale() + scaleFactor;
1180     affineTransform.scale( scaleFactor, scaleFactor );
         zTransformGroup.setTransform( affineTransform );
1182  // ZTransformGroup.animate( zTransformGroup, affineTransform, 1000, getDrawingSurface() );
      }
1184  /**
       * Insert the method's description here.
1186   * Creation date: (31-07-00 19:43:21)
       * @param newBoundsBeforeResizing edu.umd.cs.jazz.util.ZBounds
1188   */
      public void setBoundsBeforeResizing(edu.umd.cs.jazz.util.ZBounds newBoundsBeforeResizing) {
1190     boundsBeforeResizing = newBoundsBeforeResizing;
      }
1192  /**
       * Insert the method's description here.
1194   * Creation date: (28-07-00 00:50:07)
       * @param newCurrentlyClickedNode edu.umd.cs.jazz.ZVisualLeaf
1196   */
      public void setCurrentlyClickedNode(edu.umd.cs.jazz.ZVisualLeaf newCurrentlyClickedNode) {
1198     currentlyClickedNode = newCurrentlyClickedNode;
      }
1200  /**
       * Insert the method's description here.
1202   * Creation date: (08-07-00 01:30:08)
       * @param newDocuments java.util.Hashtable
1204   */
      public void setDocuments(java.util.Hashtable newDocuments) {
1206     documents = newDocuments;
      }
1208  /**
       * Insert the method's description here.
1210   * Creation date: (14-10-00 13:35:55)
       * @param newIvjArrangeAllItem javax.swing.JMenuItem
1212   */
      void setIvjArrangeAllItem(javax.swing.JMenuItem newIvjArrangeAllItem) {
1214     ivjArrangeAllItem = newIvjArrangeAllItem;
      }
1216  /**
       * Insert the method's description here.
1218   * Creation date: (14-10-00 09:53:08)
       * @param newIvjAutoLockAllItem javax.swing.JCheckBoxMenuItem
1220   */
      void setIvjAutoLockAllItem(javax.swing.JCheckBoxMenuItem newIvjAutoLockAllItem) {
```

```
1222      ivjAutoLockAllItem = newIvjAutoLockAllItem;
      }
1224  /**
       * Insert the method's description here.
1226   * Creation date: (14-10-00 13:42:49)
       * @param newIvjLockItem javax.swing.JCheckBoxMenuItem
1228   */
      void setIvjLockItem(javax.swing.JCheckBoxMenuItem newIvjLockItem) {
1230      ivjLockItem = newIvjLockItem;
      }
1232  /**
       * Insert the method's description here.
1234   * Creation date: (14-10-00 10:14:35)
       * @param newIvjSeparator3 javax.swing.JSeparator
1236   */
      void setIvjSeparator3(javax.swing.JSeparator newIvjSeparator3) {
1238      ivjSeparator3 = newIvjSeparator3;
      }
1240  /**
       * Insert the method's description here.
1242   * Creation date: (14-10-00 10:14:58)
       * @param newIvjSeparator4 javax.swing.JSeparator
1244   */
      void setIvjSeparator4(javax.swing.JSeparator newIvjSeparator4) {
1246      ivjSeparator4 = newIvjSeparator4;
      }
1248  /**
       * Insert the method's description here.
1250   * Creation date: (14-10-00 09:51:30)
       * @param newIvjUnlockAllItem javax.swing.JMenuItem
1252   */
      void setIvjUnlockAllItem(javax.swing.JMenuItem newIvjUnlockAllItem) {
1254      ivjUnlockAllItem = newIvjUnlockAllItem;
      }
1256  /**
       * Insert the method's description here.
1258   * Creation date: (29-08-00 01:03:17)
       * @param newLayoutManager clientapp.ClientPanoramaLayout
1260   */
      public void setLayoutManager(ClientPanoramaLayoutManager newLayoutManager) {
1262      layoutManager = newLayoutManager;
      }
1264  /**
       * Insert the method's description here.
1266   * Creation date: (26-07-00 11:38:26)
       * @param newLocalDocuments java.util.Hashtable
1268   */
      public void setLocalDocuments(java.util.Hashtable newLocalDocuments) {
1270      localDocuments = newLocalDocuments;
      }
1272  /**
       * Insert the method's description here.
1274   * Creation date: (29-07-00 18:22:23)
       * @param newMovingNode boolean
1276   */
      public void setMovingNode(boolean newMovingNode) {
1278      movingNode = newMovingNode;
      }
1280  /**
       * Insert the method's description here.
1282   * Creation date: (18-06-00 17:41:38)
       * @param newOwner clientapp.ClientApplication
1284   */
      public void setOwner(ClientApplication newOwner) {
1286      owner = newOwner;
      }
```

```
1288  /**
       * Insert the method's description here.
1290   * Creation date: (07-09-00 21:16:49)
       * @param newPanoramaLock boolean
1292   */
      public void setPanoramaLock(boolean newPanoramaLock) {
1294     panoramaLock = newPanoramaLock;
         PZSwing.setUpdating( newPanoramaLock );
1296  }
      /**
1298   * Insert the method's description here.
       * Creation date: (31-07-00 20:04:03)
1300   * @param newResizingNode boolean
       */
1302  public void setResizingNode(boolean newResizingNode) {
         resizingNode = newResizingNode;
1304  }
      /**
1306   * Insert the method's description here.
       * Creation date: (22-08-00 15:44:15)
1308   * @param newUpdateTimer javax.swing.Timer
       */
1310  public void setUpdateTimer(javax.swing.Timer newUpdateTimer) {
         updateTimer = newUpdateTimer;
1312  }
      /**
1314   * Insert the method's description here.
       * Creation date: (14-07-00 15:55:44)
1316   * @param zoomNode edu.umd.cs.jazz.ZNode
       */
1318  public void setZoomHandler(ZNode zoomNode)
      {
1320     zoomEventHandler = new PZoomHandler( zoomNode );
      }
1322  /**
       * Insert the method's description here.
1324   * Creation date: (29-07-00 19:19:43)
       * @param newZoomingNode boolean
1326   */
      public void setZoomingNode(boolean newZoomingNode) {
1328     zoomingNode = newZoomingNode;
      }
1330  /**
       * Insert the method's description here.
1332   * Creation date: (14-10-00 10:38:24)
       * @param node edu.umd.cs.jazz.ZVisualLeaf
1334   */
      public void unlock(ZNode node)
1336  {
         node.putClientProperty( ClientConstants.getNODE_LOCKED(), ClientConstants.getFALSE() );
1338  }
      /**
1340   * Insert the method's description here.
       * Creation date: (14-10-00 10:38:45)
1342   */
      public void unLockAll()
1344  {
         Iterator iterator = getDocuments().values().iterator();
1346     while ( iterator.hasNext() )
         {
1348        unlock( (ZVisualLeaf) iterator.next() );
         }
1350     iterator = getLocalDocuments().values().iterator();
         while ( iterator.hasNext() )
1352     {
            unlock( (ZVisualLeaf) iterator.next() );
```

```
1354     }
     }
1356 /**
      * Apply a difference array received from another group member to a document in the panorama.
1358  * Creation date: (21-08-00 17:42:03)
      * @param fqPath java.lang.String
1360  * @param newRawFileContents byte[]
      */
1362 public synchronized void updateDistributedDocument(String fqPath, byte[] differenceBitmap )
     {
1364    byte[] oldRawFileContents = null;
        byte[] newRawFileContents = null;
1366    ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
        PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
1368    JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
        java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
1370    if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
        {
1372       oldRawFileContents = (byte[]) affectedNode.getClientProperty( ClientConstants.getRAW_BITMAP() );
           newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
                differenceBitmap );
1374       affectedScrollPane.getViewport().setView( new JLabel( new ImageIcon( newRawFileContents )));
           affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
1376    }
        else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1378    {
           oldRawFileContents = ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView
                () ).getText().getBytes();
1380       newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
                differenceBitmap );
           ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
                String( newRawFileContents ));
1382    }
        affectedScrollPane.setColumnHeaderView( newDocumentLabel(
1384       ClientConstants.getDOCUMENT_LABEL(  (String) affectedNode.getClientProperty( ClientConstants.
                getDOCUMENT_NAME() ),
                                   (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
                                         ),
1386                               newRawFileContents.length,
                                   new java.util.Date() )));
1388    // ensure that the visible rectangle is unchanged
        affectedScrollPane.scrollRectToVisible( visibleRectangle );
1390    revalidate();
        getDrawingSurface().repaint();
1392
     }
1394  /**
      * Variation on updateDistributedDocument(String, byte[])
1396  * Creation date: (21-08-00 17:42:03)
      * @param fqPath java.lang.String
1398  * @param newRawFileContents byte[]
      */
1400 public synchronized void updateDistributedDocument(String senderName, String documentName, byte[]
          differenceBitmap )
     {
1402    byte[] oldRawFileContents = null;
        byte[] newRawFileContents = null;
1404    String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
        ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
1406    // the update was of a document that is not present in the panorama. It should therefore be requested
             again from
        // the update. This is now likely to succeed as the successful arrival of the update indicates that
             the connection
1408    // is "live"
        if ( affectedNode == null )
1410    {
```

```
        return; // TO DO
1412    }

1414    PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
        JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
1416    java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
        if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
1418    {
          oldRawFileContents = (byte[]) affectedNode.getClientProperty( ClientConstants.getRAW_BITMAP() );
1420      newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
              differenceBitmap );
          affectedScrollPane.getViewport().setView( new JLabel( new ImageIcon( newRawFileContents )));
1422      affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
        }
1424    else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
        {
1426      oldRawFileContents = ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView
              () ).getText().getBytes();
          newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
              differenceBitmap );
1428      ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
              String( newRawFileContents ));
        }
1430    affectedScrollPane.setColumnHeaderView( newDocumentLabel(
          ClientConstants.getDOCUMENT_LABEL(  (String) affectedNode.getClientProperty( ClientConstants.
              getDOCUMENT_NAME() ),
1432                              (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
                                      ),
                                  newRawFileContents.length,
1434                              new java.util.Date() )));
        // ensure that the visible rectangle is unchanged
1436    affectedScrollPane.scrollRectToVisible( visibleRectangle );
        revalidate();
1438    getDrawingSurface().repaint();

1440  }
      /**
1442   * Update the local documents displayed in the panorama as necessary.
       * Creation date: (21-08-00 12:11:47)
1444   */
      public synchronized void updateDocuments()
1446  {
        // safe because of short-circuit evaluation (latter part is evaluated only if the former is true)
1448    if ( ( getLocalDocuments() != null )&& ( getLocalDocuments().size() > 0 ))
        {
1450      Collection values = getLocalDocuments().values();
          Iterator iterator = values.iterator();
1452      // only notify the user that updating is taking place if the frequency is not too great to avoid
              visual clutter
          if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ))
1454      {
            owner.initMessageArea( 1, values.size(), 1, ClientConstants.getUPDATING_DOCUMENTS() );
1456      }
          while ( iterator.hasNext() )
1458      {
            ZVisualLeaf documentNode = (ZVisualLeaf) iterator.next();
1460        File file = new File( (String) documentNode.getClientProperty( ClientConstants.getDOCUMENT_NAME
                () ));
            long lastModified = ( (Long) documentNode.getClientProperty( ClientConstants.getLAST_MODIFIED()
                  )).longValue();
1462        if ( file.lastModified() != lastModified )
              {
1464          documentNode.putClientProperty( ClientConstants.getLAST_MODIFIED(), new Long( file.
                  lastModified() ));
              JScrollPane documentPane = (JScrollPane) ( (PZSwing) documentNode.getVisualComponent() ).
                  getComponent();
```

```
1466            byte rawFileContents[] = null;

1468            if ( documentPane.getViewport().getView() instanceof JLabel )
                {
1470               rawFileContents = (byte[]) documentPane.getClientProperty( ClientConstants.getRAW_BITMAP()
                        );
                }
1472            else if ( documentPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
                {
1474               rawFileContents = ( (javax.swing.text.JTextComponent )documentPane.getViewport().getView()
                        ).getText().getBytes();
                }
1476            try
                {
1478               byte newRawFileContents[] = owner.getClientManager().readDocument( file );
                   owner.getClientManager().updateDocument( file.getPath(),
1480                                          ClientInfo.getPreferences().getProperty( ClientConstants.
                                                 getNAME() ),
                                              rawFileContents,
1482                                          newRawFileContents );
                   updateLocalDocument( file.getPath(), newRawFileContents );
1484            }
                catch ( IOException ioe )
1486            {
                   System.err.println( ioe.toString() );
1488            }

1490         }
         if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ))
1492      {
            owner.advanceTask();
1494      }
         }
1496      if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ))
         {
1498        owner.resetMessageArea();
         }
1500   }
      }
1502  /**
       * Apply difference array to a document in the panorama put there by the user of the client calling this
            method.
1504   * Creation date: (22-08-00 01:56:16)
       * @param documentName java.lang.String
1506   * @param newRawFileContents byte[]
       */
1508  public synchronized void updateLocalDocument(String documentName, byte[] newRawFileContents)
      {
1510     String fqPath = ClientConstants.createFullyQualifiedDocumentName( ClientInfo.getPreferences().
              getProperty( ClientConstants.getNAME() ), documentName );
         ZVisualLeaf affectedNode = (ZVisualLeaf) getLocalDocuments().get( fqPath );
1512     PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
         JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
1514     if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
         {
1516        affectedScrollPane.getViewport().setView( new JLabel( new ImageIcon( newRawFileContents )));
            affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
1518     }
         else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1520     {
            ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
                 String( newRawFileContents ));
1522     }
         affectedScrollPane.setColumnHeaderView( newDocumentLabel(
1524        ClientConstants.getDOCUMENT_LABEL(  (String) affectedNode.getClientProperty( ClientConstants.
                 getDOCUMENT_NAME() ),
```

```
                                (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
                                      ),
1526                            newRawFileContents.length,
                                new java.util.Date() )));
1528    revalidate();
        getDrawingSurface().repaint();
1530  }
      /**
1532   * Creation date: (23-06-00 21:44:36)
       */
1534  public void validateLayout()
      {
1536    getLayoutManager().doLayout( getAllDocumentsAsArray() );
        // System.out.println("validateLayout");
1538    revalidate();
        getDrawingSurface().repaint();
1540  }
      /**
1542   * Zooms and pans the panorama camera to the argument node so that it is centered in and scaled to the
           viewport.
       * Creation date: (12-07-00 17:33:31)
1544   * @param node edu.umd.cs.jazz.ZNode
       */
1546  public void zoomAndPanToNode(ZVisualLeaf node)
      {
1548    java.awt.Rectangle targetRectangle = new java.awt.Rectangle( node.getVisualComponentGlobalBounds().
            getBounds() );
        PZoomHandler.setTargetRectangle( targetRectangle );
1550    getCamera().center( targetRectangle,
                    Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
                         getANIMATION_DURATION_IN_MILLISECS() )),
1552                getDrawingSurface() );
      }
1554  /**
       * Convenience method to allow calls using the documentID rather than the node itself
1556   * Creation date: (30-08-00 14:33:32)
       * @param documentID java.lang.String
1558   */
      public void zoomAndPanToNode(String documentID)
1560  {
        if ( getLocalDocuments().containsKey( documentID ))
1562    {
          zoomAndPanToNode( (ZVisualLeaf) getLocalDocuments().get( documentID ));
1564    }
        else
1566    {
          zoomAndPanToNode( (ZVisualLeaf) getDocuments().get( documentID ));
1568    }
      }
1570  /**
       * Insert the method's description here.
1572   * Creation date: (28-07-00 01:15:23)
       */
1574  public void zoomDocument()
      {
1576    setZoomingNode( true );
        ( (PZSwing) currentlyClickedNode.getVisualComponent() ).setZoomingNode( currentlyClickedNode );
1578    boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
        previousMousePosition = null;
1580  }
      /**
1582   * Zoom and pan the camera so that the viewport contains an overview of all documents currently in the
           panorama
       * Creation date: (01-08-00 15:23:15)
1584   */
      public void zoomToOverview()
```

```
{
   if ( ClientInfo.getPreferences().getProperty( ClientConstants.getZOOM_TO_OVERVIEW_ON_UPDATE() ).
       equalsIgnoreCase( ClientConstants.getZOOM_TO_OVERVIEW() ))
   {
      if ( (getDocuments().size() + getLocalDocuments().size()) > 0 )
      {
         java.awt.Rectangle targetRectangle = computeOverviewRectangle( getLayer().getChildren() );
         PZoomHandler.setTargetRectangle( targetRectangle );
         getCamera().center( targetRectangle,
                     Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
                         getANIMATION_DURATION_IN_MILLISECS() )),
                     getDrawingSurface() );
         owner.getClientManager().clearTopicTree();
      }
   }

}
}
```

Listing C.8: ClientPanoramaLayoutManager.java

```
package clientapp;

import edu.umd.cs.jazz.*;
import edu.umd.cs.jazz.event.*;
import java.util.*;
/**
 * The abstract root of the layout manager class hierarchy.
 * Creation date: (12-10-00 12:36:38)
 * @author:
 */
public abstract class ClientPanoramaLayoutManager implements ZLayoutManager {
   private final static java.lang.String DOCUMENT_WIDTH = "Document␣width";
   private final static java.lang.String DOCUMENT_HEIGHT = "Document␣height";
   private final static int DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
   private final static int DOCUMENT_WIDTH_DEFAULT_VALUE = 500;
/**
 * ClientPanoramaLayoutManager constructor comment.
 */
public ClientPanoramaLayoutManager() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 13:06:25)
 * @return java.lang.Object
 */
public Object clone() {
   return null;
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 23:47:58)
 * @param nodes edu.umd.cs.jazz.ZVisualLeaf[]
 */
public void doLayout(ZNode[] nodes)
{
}
/**
 * doLayout method comment.
 */
public void doLayout(ZGroup node) {}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 14:02:19)
 * @return java.lang.String
 */
```

```
    public final static java.lang.String getDOCUMENT_HEIGHT() {
48     return DOCUMENT_HEIGHT;
    }
50  /**
     * Insert the method's description here.
52   * Creation date: (12-10-00 14:24:13)
     * @return int
54   */
    public final static int getDOCUMENT_HEIGHT_DEFAULT_VALUE() {
56     return DOCUMENT_HEIGHT_DEFAULT_VALUE;
    }
58  /**
     * Insert the method's description here.
60   * Creation date: (12-10-00 14:01:04)
     * @return java.lang.String
62   */
    public final static java.lang.String getDOCUMENT_WIDTH() {
64     return DOCUMENT_WIDTH;
    }
66  /**
     * Insert the method's description here.
68   * Creation date: (12-10-00 14:24:44)
     * @return int
70   */
    public final static int getDOCUMENT_WIDTH_DEFAULT_VALUE() {
72     return DOCUMENT_WIDTH_DEFAULT_VALUE;
    }
74  /**
     * Insert the method's description here.
76   * Creation date: (12-10-00 16:13:41)
     * @return java.awt.Dimension
78   */
    public abstract java.awt.Dimension getDocumentSize();
80  /**
     * Insert the method's description here.
82   * Creation date: (18-10-00 19:42:00)
     * @return boolean
84   * @param node edu.umd.cs.jazz.ZGroup
     */
86  public boolean nodeLocked(ZNode node)
    {
88     if ( node.getClientProperty( ClientConstants.getNODE_LOCKED() )!= null )
       {
90        return node.getClientProperty( ClientConstants.getNODE_LOCKED() ).equals( ClientConstants.getTRUE
              () )? true : false;
       }
92     return false;
    }
94  /**
     * postLayout method comment.
96   */
    public void postLayout(ZGroup node) {}
98  /**
     * preLayout method comment.
100  */
    public void preLayout(ZGroup node) {}
102 /**
     * Non-overridable method to position individual node. Any subclasses should use this method rather than
          roll their own
104  * in order to hide non-essential panorama implementation detail.
     * Creation date: (14-10-00 13:19:05)
106  * @param node edu.umd.cs.jazz.ZNode
     * @param xCoordinate float
108  * @param yCoordinate float
     */
110 public final boolean setPosition(ZNode node, float xCoordinate, float yCoordinate)
```

```
{
   // System.out.println( node.getClientProperty( ClientConstants.getNODE_LOCKED() ));
   if ( nodeLocked( node )== false )
   {
      node.editor().getTransformGroup().setTranslation( xCoordinate, yCoordinate );
      return true;
   }
   else
   {
      return false;
   }
}
/**
 * Insert the method's description here.
 * Creation date: (14-10-00 13:23:49)
 * @param node edu.umd.cs.jazz.ZNode
 * @param position java.awt.Dimension
 */
public final boolean setPosition(ZNode node, java.awt.Point position)
{
   return setPosition( node, (float) position.getX(), (float) position.getY() );
}
/**
 * The newSettings argument is a set of manager specific <key,value> pairs that define the values to be
     displayed in the
 * customization box.
 * Creation date: (12-10-00 13:13:06)
 * @param newSettings java.util.Properties
 */
public void setSettings(java.util.Properties newSettings) {

}
/**
 * setState method comment.
 */
public void setState(String fieldType, String fieldName, Object fieldValue) {}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 12:58:42)
 */
public void show()
{
   show( null );
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 12:56:45)
 * @param owner javax.swing.JDialog
 */
public void show(javax.swing.JDialog owner) {}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 15:04:17)
 * @return java.util.Properties
 * @param currentSettings java.util.Properties
 */
public abstract Properties updateSettings(Properties currentSettings);
/**
 * writeObject method comment.
 */
public void writeObject(edu.umd.cs.jazz.io.ZObjectOutputStream out) throws java.io.IOException {}
/**
 * writeObjectRecurse method comment.
 */
public void writeObjectRecurse(edu.umd.cs.jazz.io.ZObjectOutputStream out) throws java.io.IOException
{}
```

```
}
```

Listing C.9: ComposeNewMessage.java

```
package clientapp;

 import javax.swing.event.*;
 import java.awt.event.*;
 import peerviewmisc.*;
 import javax.swing.JMenuItem;
 import java.awt.datatransfer.*;

 /**
 * This class implements the dialog box for composing new mesages
 * Creation date: (10-07-00 23:54:05)
 * @author:
 */
public class ComposeNewMessage extends javax.swing.JDialog
{
   public class PopUpActionListener implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         if ( ae.getSource() instanceof JMenuItem )
         {
            JMenuItem menuItem = (JMenuItem) ae.getSource();

            if ( menuItem == getPasteTextItem() )
            {
               pasteText();
            }
         }
      }
   };

   public class MouseHandler extends MouseAdapter
   {
      public void mouseClicked( MouseEvent me )
      {
         // if right-click
         if ( me.isMetaDown() )
         {
            popUpPopUpMenu( me.getPoint() );
         }
      }
      public void mouseEntered( MouseEvent me )
      {
         popDownPopUpMenu();
      }
   };

   public class PopUpMouseHandler extends MouseAdapter
   {
      public void mouseExited( MouseEvent me )
      {
         popDownPopUpMenu();
      }

      public void mouseEntered( MouseEvent me )
      {
         popDownPopUpMenu();
      }
   };

   public class BoxMouseHandler extends MouseMotionAdapter
   {
```

```
64        public void mouseMoved( MouseEvent me )
          {
66           popDownPopUpMenu();
          }
68     };

70     public class SubmitButtonHandler implements ActionListener
       {
72        public void actionPerformed( ActionEvent ae )
          {
74           submitAndClose();
          }
76     }

78     public class DiscardButtonHandler implements ActionListener
       {
80        public void actionPerformed( ActionEvent ae )
          {
82           discardAndClose();
          }
84     }
       private javax.swing.JLabel ivjAuthorLabel = null;
86     private javax.swing.JLabel ivjDateLabel = null;
       private javax.swing.JButton ivjDiscardButton = null;
88     private javax.swing.JPanel ivjJDialogContentPane = null;
       private javax.swing.JButton ivjSubmitButton = null;
90     private javax.swing.JTextField ivjAuthorTextField = null;
       private javax.swing.JEditorPane ivjContentsField = null;
92     private javax.swing.JTextField ivjDateField = null;
       private javax.swing.JPanel ivjButtonPanel = null;
94     private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
       private javax.swing.JScrollPane ivjContentsPanel = null;
96     private javax.swing.JPanel ivjHeaderPanel = null;
       private javax.swing.JTextField ivjTitleField = null;
98     private javax.swing.JLabel ivjTitleLabel = null;
       private java.util.Date date = null;
100    private peerviewmisc.Message message;
       private javax.swing.JMenuItem ivjPasteTextItem = null;
102    private javax.swing.JPopupMenu ivjPopUpMenu = null;
       private ClientApplication clientApplication = null;
104 /**
     * ComposeNewMessage constructor comment.
106  */
    public ComposeNewMessage() {
108    super();
       initialize();
110 }
    /**
112  * ComposeNewMessage constructor comment.
     * @param owner java.awt.Dialog
114  */
    public ComposeNewMessage(java.awt.Dialog owner) {
116    super(owner);
    }
118 /**
     * ComposeNewMessage constructor comment.
120  * @param owner java.awt.Dialog
     * @param title java.lang.String
122  */
    public ComposeNewMessage(java.awt.Dialog owner, String title) {
124    super(owner, title);
    }
126 /**
     * ComposeNewMessage constructor comment.
128  * @param owner java.awt.Dialog
     * @param title java.lang.String
```

```
130   * @param modal boolean
      */
132  public ComposeNewMessage(java.awt.Dialog owner, String title, boolean modal) {
        super(owner, title, modal);
134  }
     /**
136   * ComposeNewMessage constructor comment.
      * @param owner java.awt.Dialog
138   * @param modal boolean
      */
140  public ComposeNewMessage(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
142  }
     /**
144   * ComposeNewMessage constructor comment.
      * @param owner java.awt.Frame
146   */
     public ComposeNewMessage(java.awt.Frame owner) {
148     super(owner);
     }
150  /**
      * ComposeNewMessage constructor comment.
152   * @param owner java.awt.Frame
      * @param title java.lang.String
154   */
     public ComposeNewMessage(java.awt.Frame owner, String title) {
156     super(owner, title);
     }
158  /**
      * ComposeNewMessage constructor comment.
160   * @param owner java.awt.Frame
      * @param title java.lang.String
162   * @param modal boolean
      */
164  public ComposeNewMessage(java.awt.Frame owner, String title, boolean modal) {
        super(owner, title, modal);
166  }
     /**
168   * ComposeNewMessage constructor comment.
      * @param owner java.awt.Frame
170   * @param modal boolean
      */
172  public ComposeNewMessage(java.awt.Frame owner, boolean modal) {
        super(owner, modal);
174  }
     /**
176   * Insert the method's description here.
      * Creation date: (11-07-00 01:12:03)
178   */
     public void discardAndClose()
180  {
        message = null;
182     dispose();
     }
184  /**
      * Insert the method's description here.
186   * Creation date: (11-07-00 01:19:45)
      * @return java.lang.String
188   */
     public String getAuthor() {
190     return getAuthorTextField().getText();
     }
192  /**
      * Return the AuthorLabel property value.
194   * @return javax.swing.JLabel
      */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getAuthorLabel() {
   if (ivjAuthorLabel == null) {
      try {
         ivjAuthorLabel = new javax.swing.JLabel();
         ivjAuthorLabel.setName("AuthorLabel");
         ivjAuthorLabel.setText("Author");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAuthorLabel;
}
/**
 * Return the JTextField1 property value.
 * @return javax.swing.JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getAuthorTextField() {
   if (ivjAuthorTextField == null) {
      try {
         ivjAuthorTextField = new javax.swing.JTextField();
         ivjAuthorTextField.setName("AuthorTextField");
         // user code begin {1}
         ivjAuthorTextField.setEditable( false );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAuthorTextField;
}
/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
   if (ivjButtonPanel == null) {
      try {
         ivjButtonPanel = new javax.swing.JPanel();
         ivjButtonPanel.setName("ButtonPanel");
         ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
         getButtonPanel().add(getSubmitButton(), getSubmitButton().getName());
         getButtonPanel().add(getDiscardButton(), getDiscardButton().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjButtonPanel;
}
/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
262  private java.awt.FlowLayout getButtonPanelFlowLayout() {
        java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
264     try {
           /* Create part */
266        ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
           ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
268     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
270     };
        return ivjButtonPanelFlowLayout;
272  }
     /**
274   * Insert the method's description here.
      * Creation date: (07-08-00 23:35:48)
276   * @return clientapp.ClientApplication
      */
278  public ClientApplication getClientApplication() {
        return clientApplication;
280  }
     /**
282   * Insert the method's description here.
      * Creation date: (11-07-00 01:19:58)
284   * @return java.lang.String
      */
286  public String getContents() {
        return getContentsField().getText();
288  }
     /**
290   * Return the JEditorPane1 property value.
      * @return javax.swing.JEditorPane
292   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
294  private javax.swing.JEditorPane getContentsField() {
        if (ivjContentsField == null) {
296        try {
              ivjContentsField = new javax.swing.JEditorPane();
298           ivjContentsField.setName("ContentsField");
              ivjContentsField.setBounds(0, 0, 6, 23);
300           // user code begin {1}
              ivjContentsField.addMouseListener( new MouseHandler() );
302           // user code end
           } catch (java.lang.Throwable ivjExc) {
304           // user code begin {2}
              // user code end
306           handleException(ivjExc);
           }
308     }
        return ivjContentsField;
310  }
     /**
312   * Return the JScrollPane1 property value.
      * @return javax.swing.JScrollPane
314   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
316  private javax.swing.JScrollPane getContentsPanel() {
        if (ivjContentsPanel == null) {
318        try {
              ivjContentsPanel = new javax.swing.JScrollPane();
320           ivjContentsPanel.setName("ContentsPanel");
              getContentsPanel().setViewportView(getContentsField());
322           // user code begin {1}
              getContentsPanel().setBorder( javax.swing.BorderFactory.createMatteBorder(  ClientConstants.
                   getBORDER_WIDTH(),
324                                                          ClientConstants.getBORDER_WIDTH(),
                                                             ClientConstants.getBORDER_WIDTH(),
326                                                          ClientConstants.getBORDER_WIDTH(),
```

```
                                                                    getContentsPanel().getBackground() ));
328         // user code end
         } catch (java.lang.Throwable ivjExc) {
330         // user code begin {2}
            // user code end
332         handleException(ivjExc);
         }
334      }
         return ivjContentsPanel;
336   }
      /**
338    * Insert the method's description here.
       * Creation date: (16-07-00 10:56:47)
340    * @return java.util.GregorianCalendar
       */
342   public java.util.Date getDate() {
         return date;
344   }
      /**
346    * Return the JTextField2 property value.
       * @return javax.swing.JTextField
348    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
350   private javax.swing.JTextField getDateField() {
         if (ivjDateField == null) {
352         try {
            ivjDateField = new javax.swing.JTextField();
354         ivjDateField.setName("DateField");
            // user code begin {1}
356         // set date field text string to current time as obtained by instantiating Date and converting
                it to string
            date = new java.util.Date();
358         ivjDateField.setText( date.toString() );
            ivjDateField.setEditable( false );
360         // user code end
         } catch (java.lang.Throwable ivjExc) {
362         // user code begin {2}
            // user code end
364         handleException(ivjExc);
         }
366      }
         return ivjDateField;
368   }
      /**
370    * Return the DateLabel property value.
       * @return javax.swing.JLabel
372    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
374   private javax.swing.JLabel getDateLabel() {
         if (ivjDateLabel == null) {
376         try {
            ivjDateLabel = new javax.swing.JLabel();
378         ivjDateLabel.setName("DateLabel");
            ivjDateLabel.setText("Date");
380         // user code begin {1}
            // user code end
382         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
384         // user code end
            handleException(ivjExc);
386         }
         }
388      return ivjDateLabel;
      }
390   /**
       * Return the DiscardButton property value.
```

```
392  * @return javax.swing.JButton
     */
394 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getDiscardButton() {
396    if (ivjDiscardButton == null) {
          try {
398          ivjDiscardButton = new javax.swing.JButton();
             ivjDiscardButton.setName("DiscardButton");
400          ivjDiscardButton.setMnemonic('D');
             ivjDiscardButton.setText("Discard");
402          // user code begin {1}
             ivjDiscardButton.addActionListener( new DiscardButtonHandler() );
404          // user code end
          } catch (java.lang.Throwable ivjExc) {
406          // user code begin {2}
             // user code end
408          handleException(ivjExc);
          }
410    }
       return ivjDiscardButton;
412 }
    /**
414  * Return the JPanel1 property value.
     * @return javax.swing.JPanel
416  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
418 private javax.swing.JPanel getHeaderPanel() {
       if (ivjHeaderPanel == null) {
420       try {
             ivjHeaderPanel = new javax.swing.JPanel();
422          ivjHeaderPanel.setName("HeaderPanel");
             ivjHeaderPanel.setLayout(new java.awt.GridBagLayout());
424
             java.awt.GridBagConstraints constraintsAuthorLabel = new java.awt.GridBagConstraints();
426          constraintsAuthorLabel.gridx = 0; constraintsAuthorLabel.gridy = 0;
             constraintsAuthorLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
428          constraintsAuthorLabel.insets = new java.awt.Insets(4, 4, 4, 4);
             getHeaderPanel().add(getAuthorLabel(), constraintsAuthorLabel);
430
             java.awt.GridBagConstraints constraintsDateLabel = new java.awt.GridBagConstraints();
432          constraintsDateLabel.gridx = 0; constraintsDateLabel.gridy = 1;
             constraintsDateLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
434          constraintsDateLabel.insets = new java.awt.Insets(4, 4, 4, 4);
             getHeaderPanel().add(getDateLabel(), constraintsDateLabel);
436
             java.awt.GridBagConstraints constraintsTitleLabel = new java.awt.GridBagConstraints();
438          constraintsTitleLabel.gridx = 0; constraintsTitleLabel.gridy = 2;
             constraintsTitleLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
440          constraintsTitleLabel.insets = new java.awt.Insets(4, 4, 4, 4);
             getHeaderPanel().add(getTitleLabel(), constraintsTitleLabel);
442
             java.awt.GridBagConstraints constraintsAuthorTextField = new java.awt.GridBagConstraints();
444          constraintsAuthorTextField.gridx = 1; constraintsAuthorTextField.gridy = 0;
             constraintsAuthorTextField.fill = java.awt.GridBagConstraints.BOTH;
446          constraintsAuthorTextField.anchor = java.awt.GridBagConstraints.EAST;
             constraintsAuthorTextField.weightx = 1.0;
448          constraintsAuthorTextField.ipadx = 1;
             constraintsAuthorTextField.insets = new java.awt.Insets(4, 4, 4, 4);
450          getHeaderPanel().add(getAuthorTextField(), constraintsAuthorTextField);

452          java.awt.GridBagConstraints constraintsDateField = new java.awt.GridBagConstraints();
             constraintsDateField.gridx = 1; constraintsDateField.gridy = 1;
454          constraintsDateField.fill = java.awt.GridBagConstraints.HORIZONTAL;
             constraintsDateField.anchor = java.awt.GridBagConstraints.EAST;
456          constraintsDateField.weightx = 1.0;
             constraintsDateField.insets = new java.awt.Insets(4, 4, 4, 4);
```

```
458         getHeaderPanel().add(getDateField(), constraintsDateField);

460         java.awt.GridBagConstraints constraintsTitleField = new java.awt.GridBagConstraints();
            constraintsTitleField.gridx = 1; constraintsTitleField.gridy = 2;
462         constraintsTitleField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsTitleField.weightx = 1.0;
464         constraintsTitleField.insets = new java.awt.Insets(4, 4, 4, 4);
            getHeaderPanel().add(getTitleField(), constraintsTitleField);
466         // user code begin {1}
            getHeaderPanel().setBorder( javax.swing.BorderFactory.createMatteBorder( ClientConstants.
                 getBORDER_WIDTH(),
468                                                          ClientConstants.getBORDER_WIDTH(),
                                                             ClientConstants.getBORDER_WIDTH(),
470                                                          ClientConstants.getBORDER_WIDTH(),
                                                             getHeaderPanel().getBackground() ));
472         // user code end
         } catch (java.lang.Throwable ivjExc) {
474         // user code begin {2}
            // user code end
476         handleException(ivjExc);
         }
478   }
      return ivjHeaderPanel;
480 }
    /**
482  * Return the JDialogContentPane property value.
     * @return javax.swing.JPanel
484  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
486 private javax.swing.JPanel getJDialogContentPane() {
      if (ivjJDialogContentPane == null) {
488      try {
            ivjJDialogContentPane = new javax.swing.JPanel();
490         ivjJDialogContentPane.setName("JDialogContentPane");
            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
492         getJDialogContentPane().add(getHeaderPanel(), "North");
            getJDialogContentPane().add(getContentsPanel(), "Center");
494         getJDialogContentPane().add(getButtonPanel(), "South");
            // user code begin {1}
496         // user code end
         } catch (java.lang.Throwable ivjExc) {
498         // user code begin {2}
            // user code end
500         handleException(ivjExc);
         }
502   }
      return ivjJDialogContentPane;
504 }
    /**
506  * Insert the method's description here.
     * Creation date: (18-07-00 08:37:59)
508  * @return peerviewmisc.Message
     */
510 public peerviewmisc.Message getMessage() {
      return message;
512 }
    /**
514  * Return the PasteTextItem property value.
     * @return javax.swing.JMenuItem
516  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
518 private javax.swing.JMenuItem getPasteTextItem() {
      if (ivjPasteTextItem == null) {
520      try {
            ivjPasteTextItem = new javax.swing.JMenuItem();
522         ivjPasteTextItem.setName("PasteTextItem");
```

```
            ivjPasteTextItem.setMnemonic('P');
524         ivjPasteTextItem.setText("Paste␣text");
            // user code begin {1}
526         ivjPasteTextItem.addActionListener( new PopUpActionListener() );
            // user code end
528      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
530         // user code end
            handleException(ivjExc);
532      }
         }
534      return ivjPasteTextItem;
      }
536   /**
       * Return the PopUpMenu property value.
538    * @return javax.swing.JPopupMenu
       */
540   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JPopupMenu getPopUpMenu() {
542      if (ivjPopUpMenu == null) {
            try {
544         ivjPopUpMenu = new javax.swing.JPopupMenu();
            ivjPopUpMenu.setName("PopUpMenu");
546         ivjPopUpMenu.add(getPasteTextItem());
            // user code begin {1}
548         // user code end
         } catch (java.lang.Throwable ivjExc) {
550         // user code begin {2}
            // user code end
552         handleException(ivjExc);
         }
554      }
         return ivjPopUpMenu;
556   }
      /**
558    * Return the SubmitButton property value.
       * @return javax.swing.JButton
560    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
562   private javax.swing.JButton getSubmitButton() {
         if (ivjSubmitButton == null) {
564         try {
            ivjSubmitButton = new javax.swing.JButton();
566         ivjSubmitButton.setName("SubmitButton");
            ivjSubmitButton.setMnemonic('s');
568         ivjSubmitButton.setText("Submit");
            // user code begin {1}
570         ivjSubmitButton.addActionListener( new SubmitButtonHandler() );
            // user code end
572         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
574         // user code end
            handleException(ivjExc);
576         }
         }
578      return ivjSubmitButton;
      }
580   /**
       * Insert the method's description here.
582    * Creation date: (11-07-00 01:25:12)
       * @return java.lang.String
584    */
      public String getTitle() {
586      return getTitleField().getText();
      }
588   /**
```

```
     * Return the JTextField3 property value.
590  * @return javax.swing.JTextField
     */
592  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JTextField getTitleField() {
594     if (ivjTitleField == null) {
           try {
596           ivjTitleField = new javax.swing.JTextField();
              ivjTitleField.setName("TitleField");
598           // user code begin {1}
              // user code end
600        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
602           // user code end
              handleException(ivjExc);
604        }
        }
606     return ivjTitleField;
     }
608  /**
     * Return the Subject property value.
610  * @return javax.swing.JLabel
     */
612  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JLabel getTitleLabel() {
614     if (ivjTitleLabel == null) {
           try {
616           ivjTitleLabel = new javax.swing.JLabel();
              ivjTitleLabel.setName("TitleLabel");
618           ivjTitleLabel.setText("Title");
              // user code begin {1}
620           // user code end
           } catch (java.lang.Throwable ivjExc) {
622           // user code begin {2}
              // user code end
624           handleException(ivjExc);
           }
626     }
        return ivjTitleLabel;
628  }
     /**
630  * Called whenever the part throws an exception.
     * @param exception java.lang.Throwable
632  */
     private void handleException(java.lang.Throwable exception) {
634
        /* Uncomment the following lines to print uncaught exceptions to stdout */
636     // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
638  }
     /**
640  * Initialize the class.
     */
642  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
644     try {
           // user code begin {1}
646        // user code end
           setName("ComposeNewMessage");
648        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setSize(633, 490);
650        setModal(true);
           setTitle("Compose␣new␣message");
652        setContentPane(getJDialogContentPane());
        } catch (java.lang.Throwable ivjExc) {
654        handleException(ivjExc);
```

```
      }
656   // user code begin {2}
      // default positioning puts the dialog at center of screen. The method used is a slight hack, but
          works fine.
658   setLocationRelativeTo( getOwner() );
      // user code end
660 }
    /**
662  * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
664  */
    public static void main(java.lang.String[] args) {
666   try {
         ComposeNewMessage aComposeNewMessage;
668      aComposeNewMessage = new ComposeNewMessage();
         aComposeNewMessage.setModal(true);
670      aComposeNewMessage.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
672            System.exit(0);
            };
674      });
         aComposeNewMessage.setVisible(true);
676   } catch (Throwable exception) {
         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
678      exception.printStackTrace(System.out);
      }
680 }
    /**
682  * Paste the contents of the clipboard, if any, into the message at the current caret position.
     * Creation date: (07-08-00 23:31:17)
684  */
    public void pasteText()
686 {
      getContentsField().paste();
688 }
    /**
690  * Insert the method's description here.
     * Creation date: (08-08-00 10:03:32)
692  */
    public void popDownPopUpMenu()
694 {
      if ( getPopUpMenu().isVisible() )
696   {
         getPopUpMenu().setVisible( false );
698   }
    }
700 /**
     * Display the pop-up menu.
702  * Creation date: (07-08-00 23:20:22)
     */
704 public void popUpPopUpMenu( java.awt.Point position )
    {
706   getPopUpMenu().show( getContentsPanel(), (int) position.getX(), (int) position.getY() );
    }
708 /**
     * Insert the method's description here.
710  * Creation date: (11-07-00 00:38:35)
     * @param newAuthorName java.lang.String
712  */
    public void setAuthor(String newAuthorName)
714 {
      getAuthorTextField().setText( newAuthorName );
716 }
    /**
718  * Insert the method's description here.
     * Creation date: (07-08-00 23:35:48)
```

```
 * @param newClientApplication clientapp.ClientApplication
 */
public void setClientApplication(ClientApplication newClientApplication) {
   clientApplication = newClientApplication;
}
/**
 * Insert the method's description here.
 * Creation date: (16-07-00 10:56:47)
 * @param newDate java.util.GregorianCalendar
 */
public void setDate(java.util.Date newDate) {
   date = newDate;
}
/**
 * Insert the method's description here.
 * Creation date: (18-07-00 08:37:59)
 * @param newMessage peerviewmisc.Message
 */
public void setMessage(peerviewmisc.Message newMessage) {
   message = newMessage;
}
/**
 * Package the contents of the message box fields into a message data structure and then dispose of the
     box itself.
 * The message structure is part of the dialog box object and can be retrieved by the caller after the
     box is disposed of.
 * Creation date: (11-07-00 01:11:45)
 */
public void submitAndClose()
{
   if ( getTitleField().getText().length() == 0 )
   {
      MessageBox messageBox = new MessageBox();
      messageBox.setText( ClientConstants.getTITLE_REQUIRED_MESSAGE() );
      messageBox.show();
      return;
   }
   message = new peerviewmisc.Message();

   message.setAuthor( getAuthor() );
   message.setContents( getContents() + ClientConstants.getCONTENTS_AND_SIGNATURE_DIVIDER() + ClientInfo
        .getPreferences().getProperty( ClientConstants.getSIGNATURE() ));
   message.setCreationDate( getDate() );
   message.setTitle( getTitle() );
   dispose();
}
}
```

Listing C.10: DiscussionFrame.java

```
package clientapp;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import javax.swing.tree.*;
/**
 * This class implements the discussion panel in the bottom portion of the client application window.
 * Creation date: (24-05-00 13:06:10)
 * @author:
 */
public class DiscussionFrame extends JFrame {
   private JMenuItem ivjAboutItem = null;
   private JMenuItem ivjCloseWindowItem = null;
   private JMenuItem ivjContentsItem = null;
   private JButton ivjDefaultToolBarButton = null;
```

```
18     private JMenuBar ivjDiscussionFrameJMenuBar = null;
       private JMenu ivjDiscussionMenu = null;
20     private JTree ivjDiscussionTree = null;
       private JMenu ivjHelpMenu = null;
22     private JPanel ivjJFrameContentPane = null;
       private JPanel ivjJPanel1 = null;
24     private JPanel ivjJPanel2 = null;
       private JSeparator ivjJSeparator1 = null;
26     private JSplitPane ivjJSplitPane1 = null;
       private JToolBar ivjJToolBar1 = null;
28     private JButton ivjJToolBarButton1 = null;
       private JButton ivjJToolBarButton2 = null;
30     private JButton ivjJToolBarButton3 = null;
       private JTextPane ivjMessageDisplayArea = null;
32     private JMenuItem ivjNewMessageItem = null;
       private JMenuItem ivjNewThreadItem = null;
34     private JLabel ivjAuthor = null;
       private JLabel ivjAuthorLabel = null;
36     private JLabel ivjDateLabel = null;
       private JLabel ivjJLabel1 = null;
38     private JLabel ivjJLabel2 = null;
       private BoxLayout ivjJPanel2BoxLayout = null;
40     private JPanel ivjPanelHoldingHeader = null;
       private JPanel ivjPanelHoldingHeaderFields = null;
42     private JPanel ivjPanelHoldingHeadingLabels = null;
       private JLabel ivjSubjectLabel = null;
44     private BoxLayout ivjPanelHoldingHeaderFieldsBoxLayout = null;
       private BoxLayout ivjPanelHoldingHeadingLabelsBoxLayout = null;
46     private JLabel ivjAuthorLabel1 = null;
       private JDialog ivjComposeContribution = null;
48     private JLabel ivjDateLabel1 = null;
       private JPanel ivjJDialogContentPane = null;
50     private JPanel ivjMessagePanel = null;
       private BoxLayout ivjMessagePanelBoxLayout = null;
52     private JLabel ivjSubjectLabel1 = null;
       IvjEventHandler ivjEventHandler = new IvjEventHandler();
54     private JTextPane ivjMessageDisplayAreaOnComposeBox = null;
       private JPanel ivjPanelHoldingHeaderFieldsOnComposeBox = null;
56     private BoxLayout ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = null;
       private JPanel ivjPanelHoldingHeaderOnComposeBox = null;
58     private JPanel ivjPanelHoldingHeadingLabelsOnComposeBox = null;
       private BoxLayout ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = null;
60     private JPanel ivjJPanel3 = null;
       private FlowLayout ivjJPanel3FlowLayout = null;
62     private BoxLayout ivjJDialogContentPaneBoxLayout = null;
       private JTextField ivjJTextField1 = null;
64     private JTextField ivjJTextField11 = null;
       private JTextField ivjJTextField12 = null;
66     private JPanel ivjJPanel4 = null;
       private JButton ivjDiscardButton = null;
68     private JButton ivjSubmitButton = null;

70   class IvjEventHandler implements java.awt.event.ActionListener {
         public void actionPerformed(java.awt.event.ActionEvent e) {
72          if (e.getSource() == DiscussionFrame.this.getNewMessageItem())
               connEtoM1(e);
74          if (e.getSource() == DiscussionFrame.this.getJToolBarButton1())
               connEtoM2(e);
76       };
       };
78   /**
      * DiscussionFrame constructor comment.
80    */
     public DiscussionFrame() {
82     super();
       initialize();
```

```
}
/**
 * DiscussionFrame constructor comment.
 * @param title java.lang.String
 */
public DiscussionFrame(String title) {
   super(title);
}
/**
 * connEtoM1: (NewMessageItem.action.actionPerformed(java.awt.event.ActionEvent) --> ComposeContribution
     .show()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM1(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      getComposeContribution().show();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM2: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) -->
     ComposeContribution.show()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM2(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      getComposeContribution().show();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Return the AboutItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getAboutItem() {
   if (ivjAboutItem == null) {
      try {
         ivjAboutItem = new javax.swing.JMenuItem();
         ivjAboutItem.setName("AboutItem");
         ivjAboutItem.setText("About");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAboutItem;
```

```
148  }
     /**
150   * Return the Author property value.
      * @return javax.swing.JLabel
152   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
154  private javax.swing.JLabel getAuthor() {
        if (ivjAuthor == null) {
156        try {
              ivjAuthor = new javax.swing.JLabel();
158           ivjAuthor.setName("Author");
              ivjAuthor.setText("JLabel1");
160           ivjAuthor.setForeground(java.awt.SystemColor.controlText);
              ivjAuthor.setMinimumSize(new java.awt.Dimension(45, 23));
162           ivjAuthor.setMaximumSize(new java.awt.Dimension(450, 23));
              // user code begin {1}
164           // user code end
           } catch (java.lang.Throwable ivjExc) {
166           // user code begin {2}
              // user code end
168           handleException(ivjExc);
           }
170     }
        return ivjAuthor;
172  }
     /**
174   * Return the AuthorLabel property value.
      * @return javax.swing.JLabel
176   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
178  private javax.swing.JLabel getAuthorLabel() {
        if (ivjAuthorLabel == null) {
180        try {
              ivjAuthorLabel = new javax.swing.JLabel();
182           ivjAuthorLabel.setName("AuthorLabel");
              ivjAuthorLabel.setText("Author:");
184           ivjAuthorLabel.setMaximumSize(new java.awt.Dimension(41, 23));
              ivjAuthorLabel.setMinimumSize(new java.awt.Dimension(41, 23));
186           // user code begin {1}
              // user code end
188        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
190           // user code end
              handleException(ivjExc);
192        }
        }
194     return ivjAuthorLabel;
     }
196  /**
      * Return the AuthorLabel1 property value.
198   * @return javax.swing.JLabel
      */
200  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JLabel getAuthorLabel1() {
202     if (ivjAuthorLabel1 == null) {
           try {
204           ivjAuthorLabel1 = new javax.swing.JLabel();
              ivjAuthorLabel1.setName("AuthorLabel1");
206           ivjAuthorLabel1.setPreferredSize(new java.awt.Dimension(41, 23));
              ivjAuthorLabel1.setText("Author:");
208           ivjAuthorLabel1.setMaximumSize(new java.awt.Dimension(41, 23));
              ivjAuthorLabel1.setMinimumSize(new java.awt.Dimension(41, 23));
210           // user code begin {1}
              // user code end
212        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
```

```
214         // user code end
            handleException(ivjExc);
216      }
         }
218      return ivjAuthorLabel1;
      }
220   /**
       * Return the CloseWindowItem property value.
222    * @return javax.swing.JMenuItem
       */
224   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JMenuItem getCloseWindowItem() {
226      if (ivjCloseWindowItem == null) {
            try {
228         ivjCloseWindowItem = new javax.swing.JMenuItem();
            ivjCloseWindowItem.setName("CloseWindowItem");
230         ivjCloseWindowItem.setText("Close␣window");
            // user code begin {1}
232         // user code end
            } catch (java.lang.Throwable ivjExc) {
234         // user code begin {2}
            // user code end
236         handleException(ivjExc);
            }
238      }
         return ivjCloseWindowItem;
240   }
      /**
242    * Return the ComposeContribution property value.
       * @return javax.swing.JDialog
244    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
246   private javax.swing.JDialog getComposeContribution() {
         if (ivjComposeContribution == null) {
248         try {
            ivjComposeContribution = new javax.swing.JDialog();
250         ivjComposeContribution.setName("ComposeContribution");
            ivjComposeContribution.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
252         ivjComposeContribution.setBounds(40, 672, 564, 500);
            ivjComposeContribution.setTitle("Compose␣new␣message");
254         getComposeContribution().setContentPane(getJDialogContentPane());
            // user code begin {1}
256         // user code end
            } catch (java.lang.Throwable ivjExc) {
258         // user code begin {2}
            // user code end
260         handleException(ivjExc);
            }
262      }
         return ivjComposeContribution;
264   }
      /**
266    * Return the ContentsItem property value.
       * @return javax.swing.JMenuItem
268    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
270   private javax.swing.JMenuItem getContentsItem() {
         if (ivjContentsItem == null) {
272         try {
            ivjContentsItem = new javax.swing.JMenuItem();
274         ivjContentsItem.setName("ContentsItem");
            ivjContentsItem.setText("Contents");
276         // user code begin {1}
            // user code end
278         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
```

```
280         // user code end
            handleException(ivjExc);
282      }
         }
284      return ivjContentsItem;
      }
286   /**
       * Return the DateLabel property value.
288    * @return javax.swing.JLabel
       */
290   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JLabel getDateLabel() {
292      if (ivjDateLabel == null) {
            try {
294            ivjDateLabel = new javax.swing.JLabel();
               ivjDateLabel.setName("DateLabel");
296            ivjDateLabel.setText("Date:");
               ivjDateLabel.setMaximumSize(new java.awt.Dimension(29, 23));
298            ivjDateLabel.setMinimumSize(new java.awt.Dimension(29, 23));
               // user code begin {1}
300            // user code end
            } catch (java.lang.Throwable ivjExc) {
302            // user code begin {2}
               // user code end
304            handleException(ivjExc);
            }
306      }
         return ivjDateLabel;
308   }
      /**
310    * Return the DateLabel1 property value.
       * @return javax.swing.JLabel
312    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
314   private javax.swing.JLabel getDateLabel1() {
         if (ivjDateLabel1 == null) {
316         try {
               ivjDateLabel1 = new javax.swing.JLabel();
318            ivjDateLabel1.setName("DateLabel1");
               ivjDateLabel1.setPreferredSize(new java.awt.Dimension(29, 23));
320            ivjDateLabel1.setText("Date:");
               ivjDateLabel1.setMaximumSize(new java.awt.Dimension(29, 23));
322            ivjDateLabel1.setMinimumSize(new java.awt.Dimension(29, 23));
               // user code begin {1}
324            // user code end
            } catch (java.lang.Throwable ivjExc) {
326            // user code begin {2}
               // user code end
328            handleException(ivjExc);
            }
330      }
         return ivjDateLabel1;
332   }
      /**
334    * Return the DefaultToolBarButton property value.
       * @return javax.swing.JButton
336    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
338   private javax.swing.JButton getDefaultToolBarButton() {
         if (ivjDefaultToolBarButton == null) {
340         try {
               ivjDefaultToolBarButton = new javax.swing.JButton();
342            ivjDefaultToolBarButton.setName("DefaultToolBarButton");
               ivjDefaultToolBarButton.setText("");
344            ivjDefaultToolBarButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
               ivjDefaultToolBarButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
```

```
346         ivjDefaultToolBarButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                toolbarButtonGraphics/general/Add16.gif")));
            ivjDefaultToolBarButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
348         // user code begin {1}
            // user code end
350      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
352         // user code end
            handleException(ivjExc);
354      }
         }
356      return ivjDefaultToolBarButton;
      }
358   /**
       * Return the JButton2 property value.
360    * @return javax.swing.JButton
       */
362   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getDiscardButton() {
364      if (ivjDiscardButton == null) {
            try {
366         ivjDiscardButton = new javax.swing.JButton();
            ivjDiscardButton.setName("DiscardButton");
368         ivjDiscardButton.setText("Discard");
            // user code begin {1}
370         // user code end
            } catch (java.lang.Throwable ivjExc) {
372         // user code begin {2}
            // user code end
374         handleException(ivjExc);
            }
376      }
         return ivjDiscardButton;
378   }
      /**
380    * Return the DiscussionFrameJMenuBar property value.
       * @return javax.swing.JMenuBar
382    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
384   private javax.swing.JMenuBar getDiscussionFrameJMenuBar() {
         if (ivjDiscussionFrameJMenuBar == null) {
386         try {
            ivjDiscussionFrameJMenuBar = new javax.swing.JMenuBar();
388         ivjDiscussionFrameJMenuBar.setName("DiscussionFrameJMenuBar");
            ivjDiscussionFrameJMenuBar.add(getDiscussionMenu());
390         ivjDiscussionFrameJMenuBar.add(getHelpMenu());
            // user code begin {1}
392         // user code end
            } catch (java.lang.Throwable ivjExc) {
394         // user code begin {2}
            // user code end
396         handleException(ivjExc);
            }
398      }
         return ivjDiscussionFrameJMenuBar;
400   }
      /**
402    * Return the DiscussionMenu property value.
       * @return javax.swing.JMenu
404    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
406   private javax.swing.JMenu getDiscussionMenu() {
         if (ivjDiscussionMenu == null) {
408         try {
            ivjDiscussionMenu = new javax.swing.JMenu();
410         ivjDiscussionMenu.setName("DiscussionMenu");
```

```
            ivjDiscussionMenu.setText("Discuss");
412         ivjDiscussionMenu.add(getNewThreadItem());
            ivjDiscussionMenu.add(getNewMessageItem());
414         ivjDiscussionMenu.add(getJSeparator1());
            ivjDiscussionMenu.add(getCloseWindowItem());
416         // user code begin {1}
            // user code end
418      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
420         // user code end
            handleException(ivjExc);
422      }
         }
424      return ivjDiscussionMenu;
      }
426   /**
       * Return the DiscussionTree property value.
428    * @return javax.swing.JTree
       */
430   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTree getDiscussionTree() {
432      if (ivjDiscussionTree == null) {
            try {
434         ivjDiscussionTree = new javax.swing.JTree();
            ivjDiscussionTree.setName("DiscussionTree");
436         ivjDiscussionTree.setToolTipText("Discussion␣tree");
            ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 200));
438         ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
            // user code begin {1}
440         // user code end
            } catch (java.lang.Throwable ivjExc) {
442         // user code begin {2}
            // user code end
444         handleException(ivjExc);
            }
446      }
         return ivjDiscussionTree;
448   }
      /**
450    * Return the HelpMenu property value.
       * @return javax.swing.JMenu
452    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
454   private javax.swing.JMenu getHelpMenu() {
         if (ivjHelpMenu == null) {
456         try {
            ivjHelpMenu = new javax.swing.JMenu();
458         ivjHelpMenu.setName("HelpMenu");
            ivjHelpMenu.setText("Help");
460         ivjHelpMenu.add(getContentsItem());
            ivjHelpMenu.add(getAboutItem());
462         // user code begin {1}
            // user code end
464         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
466         // user code end
            handleException(ivjExc);
468         }
         }
470      return ivjHelpMenu;
      }
472   /**
       * Return the JDialogContentPane property value.
474    * @return javax.swing.JPanel
       */
476   /* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
    private javax.swing.JPanel getJDialogContentPane() {
478    if (ivjJDialogContentPane == null) {
          try {
480          ivjJDialogContentPane = new javax.swing.JPanel();
             ivjJDialogContentPane.setName("JDialogContentPane");
482          ivjJDialogContentPane.setLayout(getJDialogContentPaneBoxLayout());
             getJDialogContentPane().add(getMessagePanel(), getMessagePanel().getName());
484          getJDialogContentPane().add(getJPanel3(), getJPanel3().getName());
             getJDialogContentPane().add(getJPanel4(), getJPanel4().getName());
486          // user code begin {1}
             // user code end
488       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
490          // user code end
             handleException(ivjExc);
492       }
       }
494    return ivjJDialogContentPane;
    }
496 /**
     * Return the JDialogContentPaneBoxLayout property value.
498  * @return javax.swing.BoxLayout
     */
500 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.BoxLayout getJDialogContentPaneBoxLayout() {
502    javax.swing.BoxLayout ivjJDialogContentPaneBoxLayout = null;
       try {
504       /* Create part */
          ivjJDialogContentPaneBoxLayout = new javax.swing.BoxLayout(getJDialogContentPane(), javax.swing.
               BoxLayout.Y_AXIS);
506    } catch (java.lang.Throwable ivjExc) {
          handleException(ivjExc);
508    };
       return ivjJDialogContentPaneBoxLayout;
510 }
    /**
512  * Return the JFrameContentPane property value.
     * @return javax.swing.JPanel
514  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
516 private javax.swing.JPanel getJFrameContentPane() {
       if (ivjJFrameContentPane == null) {
518       try {
             ivjJFrameContentPane = new javax.swing.JPanel();
520          ivjJFrameContentPane.setName("JFrameContentPane");
             ivjJFrameContentPane.setLayout(new java.awt.BorderLayout());
522          getJFrameContentPane().add(getJToolBar1(), "North");
             getJFrameContentPane().add(getJSplitPane1(), "Center");
524          // user code begin {1}
             // user code end
526       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
528          // user code end
             handleException(ivjExc);
530       }
       }
532    return ivjJFrameContentPane;
    }
534 /**
     * Return the JLabel1 property value.
536  * @return javax.swing.JLabel
     */
538 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getJLabel1() {
540    if (ivjJLabel1 == null) {
          try {
```

```
542         ivjJLabel1 = new javax.swing.JLabel();
            ivjJLabel1.setName("JLabel1");
544         ivjJLabel1.setText("JLabel1");
            ivjJLabel1.setForeground(java.awt.SystemColor.controlText);
546         ivjJLabel1.setMaximumSize(new java.awt.Dimension(45, 23));
            ivjJLabel1.setMinimumSize(new java.awt.Dimension(45, 23));
548         // user code begin {1}
            // user code end
550      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
552         // user code end
            handleException(ivjExc);
554      }
         }
556      return ivjJLabel1;
      }
558   /**
       * Return the JLabel2 property value.
560    * @return javax.swing.JLabel
       */
562   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JLabel getJLabel2() {
564      if (ivjJLabel2 == null) {
            try {
566         ivjJLabel2 = new javax.swing.JLabel();
            ivjJLabel2.setName("JLabel2");
568         ivjJLabel2.setText("JLabel2");
            ivjJLabel2.setMaximumSize(new java.awt.Dimension(45, 23));
570         ivjJLabel2.setForeground(java.awt.SystemColor.controlText);
            ivjJLabel2.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
572         ivjJLabel2.setMinimumSize(new java.awt.Dimension(45, 23));
            // user code begin {1}
574         // user code end
            } catch (java.lang.Throwable ivjExc) {
576         // user code begin {2}
            // user code end
578         handleException(ivjExc);
            }
580      }
         return ivjJLabel2;
582   }
      /**
584    * Return the JPanel1 property value.
       * @return javax.swing.JPanel
586    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
588   private javax.swing.JPanel getJPanel1() {
         if (ivjJPanel1 == null) {
590         try {
            ivjJPanel1 = new javax.swing.JPanel();
592         ivjJPanel1.setName("JPanel1");
            ivjJPanel1.setLayout(new java.awt.BorderLayout());
594         getJPanel1().add(getDiscussionTree(), "Center");
            // user code begin {1}
596         // user code end
            } catch (java.lang.Throwable ivjExc) {
598         // user code begin {2}
            // user code end
600         handleException(ivjExc);
            }
602      }
         return ivjJPanel1;
604   }
      /**
606    * Return the JPanel2 property value.
       * @return javax.swing.JPanel
```

```
608   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
610   private javax.swing.JPanel getJPanel2() {
         if (ivjJPanel2 == null) {
612         try {
               ivjJPanel2 = new javax.swing.JPanel();
614            ivjJPanel2.setName("JPanel2");
               ivjJPanel2.setLayout(getJPanel2BoxLayout());
616            getJPanel2().add(getPanelHoldingHeader(), getPanelHoldingHeader().getName());
               getJPanel2().add(getMessageDisplayArea(), getMessageDisplayArea().getName());
618            // user code begin {1}
               // user code end
620         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
622            // user code end
               handleException(ivjExc);
624         }
         }
626      return ivjJPanel2;
      }
628   /**
       * Return the JPanel2BoxLayout property value.
630    * @return javax.swing.BoxLayout
       */
632   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.BoxLayout getJPanel2BoxLayout() {
634      javax.swing.BoxLayout ivjJPanel2BoxLayout = null;
         try {
636         /* Create part */
            ivjJPanel2BoxLayout = new javax.swing.BoxLayout(getJPanel2(), javax.swing.BoxLayout.Y_AXIS);
638      } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
640      };
         return ivjJPanel2BoxLayout;
642   }
      /**
644    * Return the JPanel3 property value.
       * @return javax.swing.JPanel
646    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
648   private javax.swing.JPanel getJPanel3() {
         if (ivjJPanel3 == null) {
650         try {
               ivjJPanel3 = new javax.swing.JPanel();
652            ivjJPanel3.setName("JPanel3");
               ivjJPanel3.setLayout(getJPanel3FlowLayout());
654            getJPanel3().add(getSubmitButton(), getSubmitButton().getName());
               getJPanel3().add(getDiscardButton(), getDiscardButton().getName());
656            // user code begin {1}
               // user code end
658         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
660            // user code end
               handleException(ivjExc);
662         }
         }
664      return ivjJPanel3;
      }
666   /**
       * Return the JPanel3FlowLayout property value.
668    * @return java.awt.FlowLayout
       */
670   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private java.awt.FlowLayout getJPanel3FlowLayout() {
672      java.awt.FlowLayout ivjJPanel3FlowLayout = null;
         try {
```

```
674         /* Create part */
            ivjJPanel3FlowLayout = new java.awt.FlowLayout();
676         ivjJPanel3FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
        } catch (java.lang.Throwable ivjExc) {
678         handleException(ivjExc);
        };
680     return ivjJPanel3FlowLayout;
    }
682 /**
     * Return the JPanel4 property value.
684  * @return javax.swing.JPanel
     */
686 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getJPanel4() {
688     if (ivjJPanel4 == null) {
            try {
690             ivjJPanel4 = new javax.swing.JPanel();
                ivjJPanel4.setName("JPanel4");
692             ivjJPanel4.setLayout(null);
                // user code begin {1}
694             // user code end
            } catch (java.lang.Throwable ivjExc) {
696             // user code begin {2}
                // user code end
698             handleException(ivjExc);
            }
700     }
        return ivjJPanel4;
702 }
    /**
704  * Return the JSeparator1 property value.
     * @return javax.swing.JSeparator
706  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
708 private javax.swing.JSeparator getJSeparator1() {
        if (ivjJSeparator1 == null) {
710         try {
                ivjJSeparator1 = new javax.swing.JSeparator();
712             ivjJSeparator1.setName("JSeparator1");
                // user code begin {1}
714             // user code end
            } catch (java.lang.Throwable ivjExc) {
716             // user code begin {2}
                // user code end
718             handleException(ivjExc);
            }
720     }
        return ivjJSeparator1;
722 }
    /**
724  * Return the JSplitPane1 property value.
     * @return javax.swing.JSplitPane
726  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
728 private javax.swing.JSplitPane getJSplitPane1() {
        if (ivjJSplitPane1 == null) {
730         try {
                ivjJSplitPane1 = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
732             ivjJSplitPane1.setName("JSplitPane1");
                ivjJSplitPane1.setDividerLocation(200);
734             getJSplitPane1().add(getJPanel1(), "top");
                getJSplitPane1().add(getJPanel2(), "bottom");
736             // user code begin {1}
                // user code end
738         } catch (java.lang.Throwable ivjExc) {
                // user code begin {2}
```

```
740         // user code end
            handleException(ivjExc);
742      }
         }
744      return ivjJSplitPane1;
      }
746   /**
       * Return the JTextField1 property value.
748    * @return javax.swing.JTextField
       */
750   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTextField getJTextField1() {
752      if (ivjJTextField1 == null) {
            try {
754            ivjJTextField1 = new javax.swing.JTextField();
               ivjJTextField1.setName("JTextField1");
756            ivjJTextField1.setPreferredSize(new java.awt.Dimension(4, 23));
               ivjJTextField1.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null,
                    null));
758            ivjJTextField1.setMaximumSize(new java.awt.Dimension(2147483647, 23));
               ivjJTextField1.setMinimumSize(new java.awt.Dimension(4, 23));
760            ivjJTextField1.setEditable(false);
               // user code begin {1}
762            // user code end
            } catch (java.lang.Throwable ivjExc) {
764            // user code begin {2}
               // user code end
766            handleException(ivjExc);
            }
768      }
         return ivjJTextField1;
770   }
      /**
772    * Return the JTextField11 property value.
       * @return javax.swing.JTextField
774    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
776   private javax.swing.JTextField getJTextField11() {
         if (ivjJTextField11 == null) {
778         try {
               ivjJTextField11 = new javax.swing.JTextField();
780            ivjJTextField11.setName("JTextField11");
               ivjJTextField11.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null,
                    null));
782            ivjJTextField11.setMaximumSize(new java.awt.Dimension(2147483647, 23));
               ivjJTextField11.setPreferredSize(new java.awt.Dimension(4, 23));
784            ivjJTextField11.setMinimumSize(new java.awt.Dimension(4, 23));
               ivjJTextField11.setEditable(false);
786            // user code begin {1}
               // user code end
788         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
790            // user code end
               handleException(ivjExc);
792         }
         }
794      return ivjJTextField11;
      }
796   /**
       * Return the JTextField12 property value.
798    * @return javax.swing.JTextField
       */
800   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTextField getJTextField12() {
802      if (ivjJTextField12 == null) {
            try {
```

```
804         ivjJTextField12 = new javax.swing.JTextField();
            ivjJTextField12.setName("JTextField12");
806         ivjJTextField12.setPreferredSize(new java.awt.Dimension(4, 23));
            ivjJTextField12.setMaximumSize(new java.awt.Dimension(2147483647, 23));
808         ivjJTextField12.setMinimumSize(new java.awt.Dimension(4, 23));
            // user code begin {1}
810         // user code end
         } catch (java.lang.Throwable ivjExc) {
812         // user code begin {2}
            // user code end
814         handleException(ivjExc);
         }
816   }
      return ivjJTextField12;
818 }
    /**
820  * Return the JToolBar1 property value.
     * @return javax.swing.JToolBar
822  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
824 private javax.swing.JToolBar getJToolBar1() {
      if (ivjJToolBar1 == null) {
826      try {
            ivjJToolBar1 = new javax.swing.JToolBar();
828         ivjJToolBar1.setName("JToolBar1");
            getJToolBar1().add(getJToolBarButton1(), getJToolBarButton1().getName());
830         ivjJToolBar1.add(getDefaultToolBarButton());
            ivjJToolBar1.addSeparator();
832         getJToolBar1().add(getJToolBarButton2(), getJToolBarButton2().getName());
            getJToolBar1().add(getJToolBarButton3(), getJToolBarButton3().getName());
834         // user code begin {1}
            // user code end
836      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
838         // user code end
            handleException(ivjExc);
840      }
      }
842   return ivjJToolBar1;
    }
844 /**
     * Return the JToolBarButton1 property value.
846  * @return javax.swing.JButton
     */
848 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getJToolBarButton1() {
850   if (ivjJToolBarButton1 == null) {
      try {
852         ivjJToolBarButton1 = new javax.swing.JButton();
            ivjJToolBarButton1.setName("JToolBarButton1");
854         ivjJToolBarButton1.setText("");
            ivjJToolBarButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
856         ivjJToolBarButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
            ivjJToolBarButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                toolbarButtonGraphics/general/New16.gif")));
858         ivjJToolBarButton1.setMargin(new java.awt.Insets(0, 0, 0, 0));
            ivjJToolBarButton1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
860         // user code begin {1}
            // user code end
862      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
864         // user code end
            handleException(ivjExc);
866      }
      }
868   return ivjJToolBarButton1;
```

```
    }
    /**
     * Return the JToolBarButton2 property value.
     * @return javax.swing.JButton
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getJToolBarButton2() {
       if (ivjJToolBarButton2 == null) {
          try {
             ivjJToolBarButton2 = new javax.swing.JButton();
             ivjJToolBarButton2.setName("JToolBarButton2");
             ivjJToolBarButton2.setText("");
             ivjJToolBarButton2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
             ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
             ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                 toolbarButtonGraphics/general/Help16.gif")));
             ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
             // user code end
          } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
             // user code end
             handleException(ivjExc);
          }
       }
       return ivjJToolBarButton2;
    }
    /**
     * Return the JToolBarButton3 property value.
     * @return javax.swing.JButton
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getJToolBarButton3() {
       if (ivjJToolBarButton3 == null) {
          try {
             ivjJToolBarButton3 = new javax.swing.JButton();
             ivjJToolBarButton3.setName("JToolBarButton3");
             ivjJToolBarButton3.setText("");
             ivjJToolBarButton3.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
             ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
             ivjJToolBarButton3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                 toolbarButtonGraphics/general/Stop16.gif")));
             ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
             // user code end
          } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
             // user code end
             handleException(ivjExc);
          }
       }
       return ivjJToolBarButton3;
    }
    /**
     * Return the MessageDisplayArea property value.
     * @return javax.swing.JTextPane
     */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JTextPane getMessageDisplayArea() {
       if (ivjMessageDisplayArea == null) {
          try {
             ivjMessageDisplayArea = new javax.swing.JTextPane();
             ivjMessageDisplayArea.setName("MessageDisplayArea");
             ivjMessageDisplayArea.setToolTipText("Message␣display␣area");
             ivjMessageDisplayArea.setEnabled(false);
             // user code begin {1}
```

```
            ivjMessageDisplayArea.setBorder(BorderFactory.createEtchedBorder());
934         // user code end
         } catch (java.lang.Throwable ivjExc) {
936         // user code begin {2}
            // user code end
938         handleException(ivjExc);
         }
940   }
      return ivjMessageDisplayArea;
942 }
    /**
944  * Return the MessageDisplayArea1 property value.
     * @return javax.swing.JTextPane
946  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
948 private javax.swing.JTextPane getMessageDisplayAreaOnComposeBox() {
      if (ivjMessageDisplayAreaOnComposeBox == null) {
950      try {
            ivjMessageDisplayAreaOnComposeBox = new javax.swing.JTextPane();
952         ivjMessageDisplayAreaOnComposeBox.setName("MessageDisplayAreaOnComposeBox");
            ivjMessageDisplayAreaOnComposeBox.setToolTipText("Message␣composition␣area");
954         ivjMessageDisplayAreaOnComposeBox.setEnabled(true);
            // user code begin {1}
956         ivjMessageDisplayAreaOnComposeBox.setBorder(BorderFactory.createEtchedBorder());
            // user code end
958      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
960         // user code end
            handleException(ivjExc);
962      }
      }
964   return ivjMessageDisplayAreaOnComposeBox;
    }
966 /**
     * Return the MessagePanel property value.
968  * @return javax.swing.JPanel
     */
970 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getMessagePanel() {
972   if (ivjMessagePanel == null) {
         try {
974         ivjMessagePanel = new javax.swing.JPanel();
            ivjMessagePanel.setName("MessagePanel");
976         ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
            ivjMessagePanel.add(getPanelHoldingHeaderOnComposeBox());
978         getMessagePanel().add(getMessageDisplayAreaOnComposeBox(), getMessageDisplayAreaOnComposeBox().
                 getName());
            // user code begin {1}
980         // user code end
         } catch (java.lang.Throwable ivjExc) {
982         // user code begin {2}
            // user code end
984         handleException(ivjExc);
         }
986   }
      return ivjMessagePanel;
988 }
    /**
990  * Return the MessagePanelBoxLayout property value.
     * @return javax.swing.BoxLayout
992  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
994 private javax.swing.BoxLayout getMessagePanelBoxLayout() {
      javax.swing.BoxLayout ivjMessagePanelBoxLayout = null;
996   try {
         /* Create part */
```

```
      ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swing.BoxLayout.
          Y_AXIS);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   };
   return ivjMessagePanelBoxLayout;
}
/**
 * Return the NewMessageItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getNewMessageItem() {
   if (ivjNewMessageItem == null) {
      try {
         ivjNewMessageItem = new javax.swing.JMenuItem();
         ivjNewMessageItem.setName("NewMessageItem");
         ivjNewMessageItem.setText("New␣message...");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjNewMessageItem;
}
/**
 * Return the NewThreadItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getNewThreadItem() {
   if (ivjNewThreadItem == null) {
      try {
         ivjNewThreadItem = new javax.swing.JMenuItem();
         ivjNewThreadItem.setName("NewThreadItem");
         ivjNewThreadItem.setText("New␣thread");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjNewThreadItem;
}
/**
 * Return the PanelHoldingHeader property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeader() {
   if (ivjPanelHoldingHeader == null) {
      try {
         ivjPanelHoldingHeader = new javax.swing.JPanel();
         ivjPanelHoldingHeader.setName("PanelHoldingHeader");
         ivjPanelHoldingHeader.setLayout(new javax.swing.BoxLayout(getPanelHoldingHeader(), javax.swing.
             BoxLayout.X_AXIS));
         ivjPanelHoldingHeader.setMaximumSize(new java.awt.Dimension(496, 120));
         ivjPanelHoldingHeader.setPreferredSize(new java.awt.Dimension(496, 70));
         ivjPanelHoldingHeader.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
         ivjPanelHoldingHeader.setMinimumSize(new java.awt.Dimension(496, 120));
```

```
            getPanelHoldingHeader().add(getPanelHoldingHeadingLabels(), getPanelHoldingHeadingLabels().
                getName());
1062        getPanelHoldingHeader().add(getPanelHoldingHeaderFields(), getPanelHoldingHeaderFields().getName
                ());
            // user code begin {1}
1064        // user code end
         } catch (java.lang.Throwable ivjExc) {
1066        // user code begin {2}
            // user code end
1068        handleException(ivjExc);
         }
1070     }
         return ivjPanelHoldingHeader;
1072  }
      /**
1074   * Return the PanelHoldingHeaderFields property value.
       * @return javax.swing.JPanel
1076   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1078  private javax.swing.JPanel getPanelHoldingHeaderFields() {
         if (ivjPanelHoldingHeaderFields == null) {
1080        try {
            ivjPanelHoldingHeaderFields = new javax.swing.JPanel();
1082        ivjPanelHoldingHeaderFields.setName("PanelHoldingHeaderFields");
            ivjPanelHoldingHeaderFields.setLayout(getPanelHoldingHeaderFieldsBoxLayout());
1084        ivjPanelHoldingHeaderFields.setMaximumSize(new java.awt.Dimension(450, 69));
            ivjPanelHoldingHeaderFields.setMinimumSize(new java.awt.Dimension(450, 69));
1086        getPanelHoldingHeaderFields().add(getAuthor(), getAuthor().getName());
            getPanelHoldingHeaderFields().add(getJLabel2(), getJLabel2().getName());
1088        getPanelHoldingHeaderFields().add(getJLabel1(), getJLabel1().getName());
            // user code begin {1}
1090        ivjPanelHoldingHeaderFields.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
                ivjPanelHoldingHeaderFields.getBackground()));
            // user code end
1092        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1094        // user code end
            handleException(ivjExc);
1096        }
         }
1098     return ivjPanelHoldingHeaderFields;
      }
1100  /**
       * Return the PanelHoldingHeaderFieldsBoxLayout property value.
1102   * @return javax.swing.BoxLayout
       */
1104  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.BoxLayout getPanelHoldingHeaderFieldsBoxLayout() {
1106     javax.swing.BoxLayout ivjPanelHoldingHeaderFieldsBoxLayout = null;
         try {
1108        /* Create part */
            ivjPanelHoldingHeaderFieldsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeaderFields(),
                javax.swing.BoxLayout.Y_AXIS);
1110     } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
1112     };
         return ivjPanelHoldingHeaderFieldsBoxLayout;
1114  }
      /**
1116   * Return the PanelHoldingHeaderFields1 property value.
       * @return javax.swing.JPanel
1118   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1120  private javax.swing.JPanel getPanelHoldingHeaderFieldsOnComposeBox() {
         if (ivjPanelHoldingHeaderFieldsOnComposeBox == null) {
1122        try {
```

```
            ivjPanelHoldingHeaderFieldsOnComposeBox = new javax.swing.JPanel();
1124        ivjPanelHoldingHeaderFieldsOnComposeBox.setName("PanelHoldingHeaderFieldsOnComposeBox");
            ivjPanelHoldingHeaderFieldsOnComposeBox.setPreferredSize(new java.awt.Dimension(470000, 75));
1126        ivjPanelHoldingHeaderFieldsOnComposeBox.setLayout(
                getPanelHoldingHeaderFieldsOnComposeBoxBoxLayout());
            ivjPanelHoldingHeaderFieldsOnComposeBox.setMaximumSize(new java.awt.Dimension(45000, 75));
1128        ivjPanelHoldingHeaderFieldsOnComposeBox.setMinimumSize(new java.awt.Dimension(450, 75));
            ivjPanelHoldingHeaderFieldsOnComposeBox.add(getJTextField1());
1130        getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField11(), getJTextField11().getName());
            getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField12(), getJTextField12().getName());
1132        // user code begin {1}
            ivjPanelHoldingHeaderFieldsOnComposeBox.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
                ivjPanelHoldingHeaderFieldsOnComposeBox.getBackground()));
1134
            // user code end
1136     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1138        // user code end
            handleException(ivjExc);
1140     }
      }
1142  return ivjPanelHoldingHeaderFieldsOnComposeBox;
   }
1144 /**
      * Return the PanelHoldingHeaderFieldsOnComposeBoxBoxLayout property value.
1146  * @return javax.swing.BoxLayout
      */
1148 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.BoxLayout getPanelHoldingHeaderFieldsOnComposeBoxBoxLayout() {
1150  javax.swing.BoxLayout ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = null;
      try {
1152     /* Create part */
         ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
             getPanelHoldingHeaderFieldsOnComposeBox(), javax.swing.BoxLayout.Y_AXIS);
1154  } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
1156  };
      return ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout;
1158 }
   /**
1160  * Return the PanelHoldingHeader1 property value.
      * @return javax.swing.JPanel
1162  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
1164 private javax.swing.JPanel getPanelHoldingHeaderOnComposeBox() {
      if (ivjPanelHoldingHeaderOnComposeBox == null) {
1166     try {
            ivjPanelHoldingHeaderOnComposeBox = new javax.swing.JPanel();
1168        ivjPanelHoldingHeaderOnComposeBox.setName("PanelHoldingHeaderOnComposeBox");
            ivjPanelHoldingHeaderOnComposeBox.setLayout(new javax.swing.BoxLayout(
                getPanelHoldingHeaderOnComposeBox(), javax.swing.BoxLayout.X_AXIS));
1170        ivjPanelHoldingHeaderOnComposeBox.setMaximumSize(new java.awt.Dimension(496000, 120));
            ivjPanelHoldingHeaderOnComposeBox.setPreferredSize(new java.awt.Dimension(4960000, 90));
1172        ivjPanelHoldingHeaderOnComposeBox.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
            ivjPanelHoldingHeaderOnComposeBox.setMinimumSize(new java.awt.Dimension(496, 120));
1174        getPanelHoldingHeaderOnComposeBox().add(getPanelHoldingHeadingLabelsOnComposeBox(),
                getPanelHoldingHeadingLabelsOnComposeBox().getName());
            getPanelHoldingHeaderOnComposeBox().add(getPanelHoldingHeaderFieldsOnComposeBox(),
                getPanelHoldingHeaderFieldsOnComposeBox().getName());
1176        // user code begin {1}
            // user code end
1178     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1180        // user code end
            handleException(ivjExc);
1182     }
```

```
      }
      return ivjPanelHoldingHeaderOnComposeBox;
   }
   /**
    * Return the PanelHoldingHeadingLabels property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getPanelHoldingHeadingLabels() {
      if (ivjPanelHoldingHeadingLabels == null) {
         try {
            ivjPanelHoldingHeadingLabels = new javax.swing.JPanel();
            ivjPanelHoldingHeadingLabels.setName("PanelHoldingHeadingLabels");
            ivjPanelHoldingHeadingLabels.setLayout(getPanelHoldingHeadingLabelsBoxLayout());
            getPanelHoldingHeadingLabels().add(getAuthorLabel(), getAuthorLabel().getName());
            getPanelHoldingHeadingLabels().add(getDateLabel(), getDateLabel().getName());
            getPanelHoldingHeadingLabels().add(getSubjectLabel(), getSubjectLabel().getName());
            // user code begin {1}
            ivjPanelHoldingHeadingLabels.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
                 ivjPanelHoldingHeadingLabels.getBackground()));
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjPanelHoldingHeadingLabels;
   }
   /**
    * Return the PanelHoldingHeadingLabelsBoxLayout property value.
    * @return javax.swing.BoxLayout
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.BoxLayout getPanelHoldingHeadingLabelsBoxLayout() {
      javax.swing.BoxLayout ivjPanelHoldingHeadingLabelsBoxLayout = null;
      try {
         /* Create part */
         ivjPanelHoldingHeadingLabelsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeadingLabels(),
              javax.swing.BoxLayout.Y_AXIS);
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      };
      return ivjPanelHoldingHeadingLabelsBoxLayout;
   }
   /**
    * Return the PanelHoldingHeadingLabels1 property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getPanelHoldingHeadingLabelsOnComposeBox() {
      if (ivjPanelHoldingHeadingLabelsOnComposeBox == null) {
         try {
            ivjPanelHoldingHeadingLabelsOnComposeBox = new javax.swing.JPanel();
            ivjPanelHoldingHeadingLabelsOnComposeBox.setName("PanelHoldingHeadingLabelsOnComposeBox");
            ivjPanelHoldingHeadingLabelsOnComposeBox.setPreferredSize(new java.awt.Dimension(65, 70));
            ivjPanelHoldingHeadingLabelsOnComposeBox.setLayout(
                 getPanelHoldingHeadingLabelsOnComposeBoxBoxLayout());
            ivjPanelHoldingHeadingLabelsOnComposeBox.setMaximumSize(new java.awt.Dimension(65, 75));
            ivjPanelHoldingHeadingLabelsOnComposeBox.setMinimumSize(new java.awt.Dimension(65, 75));
            getPanelHoldingHeadingLabelsOnComposeBox().add(getAuthorLabel1(), getAuthorLabel1().getName());
            getPanelHoldingHeadingLabelsOnComposeBox().add(getDateLabel1(), getDateLabel1().getName());
            getPanelHoldingHeadingLabelsOnComposeBox().add(getSubjectLabel1(), getSubjectLabel1().getName())
                 ;
            // user code begin {1}
```

```
1244          ivjPanelHoldingHeadingLabelsOnComposeBox.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
                  ivjPanelHoldingHeadingLabelsOnComposeBox.getBackground()));
              // user code end
1246       } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1248          // user code end
              handleException(ivjExc);
1250       }
        }
1252    return ivjPanelHoldingHeadingLabelsOnComposeBox;
     }
1254 /**
      * Return the PanelHoldingHeadingLabelsOnComposeBoxBoxLayout property value.
1256  * @return javax.swing.BoxLayout
      */
1258 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.BoxLayout getPanelHoldingHeadingLabelsOnComposeBoxBoxLayout() {
1260    javax.swing.BoxLayout ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = null;
        try {
1262       /* Create part */
           ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
                getPanelHoldingHeadingLabelsOnComposeBox(), javax.swing.BoxLayout.Y_AXIS);
1264    } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
1266    };
        return ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout;
1268 }
     /**
1270  * Return the SubjectLabel property value.
      * @return javax.swing.JLabel
1272  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1274 private javax.swing.JLabel getSubjectLabel() {
        if (ivjSubjectLabel == null) {
1276       try {
              ivjSubjectLabel = new javax.swing.JLabel();
1278          ivjSubjectLabel.setName("SubjectLabel");
              ivjSubjectLabel.setText("Subject:");
1280          ivjSubjectLabel.setMaximumSize(new java.awt.Dimension(46, 23));
              ivjSubjectLabel.setMinimumSize(new java.awt.Dimension(46, 23));
1282          // user code begin {1}
              // user code end
1284       } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
1286          // user code end
              handleException(ivjExc);
1288       }
        }
1290    return ivjSubjectLabel;
     }
1292 /**
      * Return the SubjectLabel1 property value.
1294  * @return javax.swing.JLabel
      */
1296 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JLabel getSubjectLabel1() {
1298    if (ivjSubjectLabel1 == null) {
        try {
1300       ivjSubjectLabel1 = new javax.swing.JLabel();
           ivjSubjectLabel1.setName("SubjectLabel1");
1302       ivjSubjectLabel1.setPreferredSize(new java.awt.Dimension(46, 23));
           ivjSubjectLabel1.setText("Subject:");
1304       ivjSubjectLabel1.setMaximumSize(new java.awt.Dimension(46, 23));
           ivjSubjectLabel1.setMinimumSize(new java.awt.Dimension(46, 23));
1306       // user code begin {1}
           // user code end
```

```
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjSubjectLabel1;
}
/**
 * Return the JButton1 property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getSubmitButton() {
   if (ivjSubmitButton == null) {
      try {
         ivjSubmitButton = new javax.swing.JButton();
         ivjSubmitButton.setName("SubmitButton");
         ivjSubmitButton.setText("Submit");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjSubmitButton;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
}
/**
 * Initializes connections
 * @exception java.lang.Exception The exception description.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initConnections() throws java.lang.Exception {
   // user code begin {1}
   // user code end
   getNewMessageItem().addActionListener(ivjEventHandler);
   getJToolBarButton1().addActionListener(ivjEventHandler);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
      setName("DiscussionFrame");
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setTitle("Discussion␣of␣xxx.yyy");
      setSize(600, 600);
      setJMenuBar(getDiscussionFrameJMenuBar());
      setContentPane(getJFrameContentPane());
      initConnections();
   } catch (java.lang.Throwable ivjExc) {
```

```
      handleException(ivjExc);
   }
   // user code begin {2}
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      DiscussionFrame aDiscussionFrame;
      aDiscussionFrame = new DiscussionFrame();
      aDiscussionFrame.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aDiscussionFrame.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JFrame");
      exception.printStackTrace(System.out);
   }
}
}
```

Listing C.11: GridLayout.java

```
package clientapp;


 import java.awt.Dimension;
 import java.lang.Exception;

 /**
 * Default layout class for the client panorama. Manages a rectangular grid ordering that ought to be
     easily customizable.
 * Creation date: (22-06-00 11:42:53)
 * @author:
 */
public class GridLayout extends PanoramaLayout {
   private int width;
   private int height;
/**
 * GridLayout constructor comment.
 */
public GridLayout()
{
   super();
   setHeight( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
       getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() )));
   setWidth( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
       getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() )));
}
/**
 * Convert the vertical and horizontal indices into a x,y coordinate pair designating a position in the
     panorama.
 * Creation date: (22-06-00 12:02:00)
 * @return java.awt.Dimension
 */
public java.awt.Dimension computePlacement() throws Exception
{
   Dimension newPlacement = new Dimension( (int) ( horizontalIndex * (int)(preferredDocumentSize.
       getWidth() + getXSpacing() )),
                              (int) ( verticalIndex * (int)(preferredDocumentSize.getHeight() +
                                  getYSpacing() )));
   if (horizontalIndex >= width)
```

```
34     {
          if (verticalIndex >= height)
36        {
             throw new Exception( ClientConstants.getOUT_OF_BOUNDS_EXCEPTION_TEXT() );
38        }
          else
40        {
             verticalIndex++;
42           horizontalIndex = 0;
          }
44     }
       else
46     {
          horizontalIndex++;
48     }
       return newPlacement;
50  }
    /**
52   * Insert the method's description here.
     * Creation date: (22-06-00 11:43:44)
54   * @return int
     */
56  public int getHeight() {
       return height;
58  }
    /**
60   * Compute and return the dimensions of a rectangle containing all the document nodes being laid out.
     * Creation date: (01-08-00 19:43:12)
62   * @return java.awt.Rectangle
     */
64  public java.awt.Rectangle getOverviewRectangle()
    {
66     java.awt.Rectangle newRectangle = new java.awt.Rectangle();
       if ( getVerticalIndex() == 0 )
68     {
          newRectangle.setRect( -( getXSpacing()/2 ), 0,
70           getHorizontalIndex() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ),
             getVerticalIndex() * ( getPreferredDocumentSize().getHeight() + getYSpacing() ));
72     }
       else
74     {
          newRectangle.setRect( -( getXSpacing()/2 ), 0,
76           getWidth() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ),
             getVerticalIndex() * ( getPreferredDocumentSize().getHeight() + getYSpacing() ));
78     }
       return newRectangle;
80  }
    /**
82   * Insert the method's description here.
     * Creation date: (22-06-00 11:43:29)
84   * @return int
     */
86  public int getWidth() {
       return width;
88  }
    /**
90   * Insert the method's description here.
     * Creation date: (23-06-00 22:17:36)
92   */
    public void resetLayout()
94  {
       horizontalIndex = 0;
96     verticalIndex = 0;
    }
98  /**
     * Insert the method's description here.
```

```
 * Creation date: (22-06-00 11:43:44)
 * @param newHeight int
 */
public void setHeight(int newHeight) {
   height = newHeight;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:43:29)
 * @param newWidth int
 */
public void setWidth(int newWidth) {
   width = newWidth;
}
}
```

Listing C.12: GridPanoramaLayoutManager.java

```
package clientapp;

 import java.util.ArrayList;
 import java.awt.event.*;
 import javax.swing.event.*;

 /**
 * This class implements a simple layout manager which arranges the panorama documents in matrix pattern
     .
 * Creation date: (28-08-00 23:56:47)
 * @author:
 */
public class GridPanoramaLayoutManager extends ClientPanoramaLayoutManager
{
   private javax.swing.JPanel ivjButtonPanel = null;
   private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
   private javax.swing.JButton ivjCancelButton = null;
   private javax.swing.JTextField ivjDocumentHeightField = null;
   private javax.swing.JLabel ivjDocumentHeightLabel = null;
   private javax.swing.JTextField ivjDocumentWidthField = null;
   private javax.swing.JLabel ivjDocumentWidthLabel = null;
   private javax.swing.JDialog ivjGridLayoutCustomizationBox = null;
   private javax.swing.JTextField ivjHorizontalSpacingField = null;
   private javax.swing.JLabel ivjHorizontalSpacingLabel = null;
   private javax.swing.JPanel ivjJDialogContentPane = null;
   private javax.swing.JPanel ivjMainPanel = null;
   private javax.swing.JButton ivjOKButton = null;
   private javax.swing.JTextField ivjVerticalSpacingField = null;
   private javax.swing.JLabel ivjVerticalSpacingLabel = null;
   private final static java.lang.String GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING = "Grid␣layout␣document
       ␣horizontal␣spacing";
   private final static java.lang.String GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING = "Grid␣layout␣document␣
       vertical␣spacing";
   private final static int GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE = 200;
   private final static int GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE = 200;

   public class CancelHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         setCommitted( false );
         getGridLayoutCustomizationBox().dispose();
      }
   };

   public class OKHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
```

```
            setCommitted( true );
48          getGridLayoutCustomizationBox().dispose();
         }
50    };
      private boolean committed = false;
52    private final static java.lang.String GRID_LAYOUT_DOCUMENT_HEIGHT = "Grid␣layout␣document␣height";
      private final static java.lang.String GRID_LAYOUT_DOCUMENT_WIDTH = "Grid␣layout␣document␣width";
54    private final static int GRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE = 600;
      private final static int GRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
56  /**
     * GridPanoramaLayout constructor comment.
58   */
    public GridPanoramaLayoutManager() {
60    super();
      initialize();
62  }
    /**
64   * Insert the method's description here.
     * Creation date: (12-10-00 11:50:40)
66   * @return java.lang.Object
     */
68  public Object clone() {
      return null;
70  }
    /**
72   * Computes a list of Point objects representing a grid structure on which the nodes can subsequently be
         distributed.
     */
74  public java.util.ArrayList computeCoordinates(edu.umd.cs.jazz.ZNode[] nodes)
    {
76    int xCoordinate = 0;
      int yCoordinate = 0;
78
      int xIncrement = 0;
80    int yIncrement = 0;
      // let document with the largest bounding box determine the spacing between all the documents
82    for ( int i = 0; i < nodes.length; i++)
      {
84       xIncrement = (int)Math.round( nodes[i].getBounds().getWidth() )> xIncrement ? (int)Math.round(
             nodes[i].getBounds().getWidth() ): xIncrement;
         yIncrement = (int)Math.round( nodes[i].getBounds().getHeight() )> yIncrement ? (int)Math.round(
             nodes[i].getBounds().getHeight() ): yIncrement;
86    }
      xIncrement += Integer.parseInt( getHorizontalSpacingField().getText() );
88    yIncrement += Integer.parseInt( getVerticalSpacingField().getText() );
      int sideLength = Math.round( (float) Math.ceil( Math.sqrt( (double) nodes.length )));
90    ArrayList list = new ArrayList();
      for ( int i = 0 ; i < nodes.length ; i++ )
92    {
         if ( (i % sideLength) == 0 )
94       {
            xCoordinate = 0;
96       }
         else
98       {
            xCoordinate += xIncrement;
100      }
         yCoordinate = (i / sideLength )* yIncrement;
102      list.add( new java.awt.Point( xCoordinate, yCoordinate ));
         // System.out.println( String.valueOf( xCoordinate )+ "-" + String.valueOf( yCoordinate ));
104   }
      return list;
106 }
    /**
108  * Distribute a set of nodes at positions given in the second argument list.
     * Creation date: (29-08-00 08:28:01)
```

```
110   * @param nodes edu.umd.cs.jazz.ZNode[]
      * @param coordinates java.util.ArrayList
112   */
      public void distributeCoordinates(edu.umd.cs.jazz.ZNode[] nodes, ArrayList coordinates)
114   {
         // translate as many nodes as there are coordinate pairs and leave the remainder unchanged
116      int i,j;
         for ( i = 0, j = 0 ; i < coordinates.size(); i++ )
118      {
            if ( nodeLocked( nodes[ i ] )== false )
120         {  java.awt.Point coordinatePair = (java.awt.Point) coordinates.get(j);
               setPosition( nodes[i], coordinatePair );
122            j++;
            }
124      }
      }
126   /**
      * Insert the method's description here.
128   * Creation date: (12-10-00 23:50:36)
      * @param nodes edu.umd.cs.jazz.ZNode[]
130   */
      public void doLayout(edu.umd.cs.jazz.ZNode[] nodes)
132   {
         // System.out.println("doLayout");
134      distributeCoordinates( nodes, computeCoordinates( nodes ));
      }
136   /**
      * Insert the method's description here.
138   * Creation date: (12-10-00 11:56:01)
      * @param node edu.umd.cs.jazz.ZGroup
140   */
      public void doLayout(edu.umd.cs.jazz.ZGroup node)
142   {

144   }
      /**
146   * Return the ButtonPanel property value.
      * @return javax.swing.JPanel
148   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
150   private javax.swing.JPanel getButtonPanel() {
         if (ivjButtonPanel == null) {
152         try {
               ivjButtonPanel = new javax.swing.JPanel();
154            ivjButtonPanel.setName("ButtonPanel");
               ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
156            getButtonPanel().add(getOKButton(), getOKButton().getName());
               getButtonPanel().add(getCancelButton(), getCancelButton().getName());
158            // user code begin {1}
               // user code end
160         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
162            // user code end
               handleException(ivjExc);
164         }
         }
166      return ivjButtonPanel;
      }
168   /**
      * Return the ButtonPanelFlowLayout property value.
170   * @return java.awt.FlowLayout
      */
172   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private java.awt.FlowLayout getButtonPanelFlowLayout() {
174      java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
         try {
```

```
176        /* Create part */
           ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
178        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
        } catch (java.lang.Throwable ivjExc) {
180        handleException(ivjExc);
        };
182     return ivjButtonPanelFlowLayout;
     }
184  /**
      * Return the CancelButton property value.
186   * @return javax.swing.JButton
      */
188  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getCancelButton() {
190     if (ivjCancelButton == null) {
           try {
192           ivjCancelButton = new javax.swing.JButton();
              ivjCancelButton.setName("CancelButton");
194           ivjCancelButton.setMnemonic('C');
              ivjCancelButton.setText("Cancel");
196           // user code begin {1}
              ivjCancelButton.addActionListener( new CancelHandler() );
198           // user code end
           } catch (java.lang.Throwable ivjExc) {
200           // user code begin {2}
              // user code end
202           handleException(ivjExc);
           }
204     }
        return ivjCancelButton;
206  }
     /**
208   * Return the DocumentHeightField property value.
      * @return javax.swing.JTextField
210   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
212  private javax.swing.JTextField getDocumentHeightField() {
        if (ivjDocumentHeightField == null) {
214        try {
              ivjDocumentHeightField = new javax.swing.JTextField();
216           ivjDocumentHeightField.setName("DocumentHeightField");
              // user code begin {1}
218           // user code end
           } catch (java.lang.Throwable ivjExc) {
220           // user code begin {2}
              // user code end
222           handleException(ivjExc);
           }
224     }
        return ivjDocumentHeightField;
226  }
     /**
228   * Return the DocumentHeightLabel property value.
      * @return javax.swing.JLabel
230   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
232  private javax.swing.JLabel getDocumentHeightLabel() {
        if (ivjDocumentHeightLabel == null) {
234        try {
              ivjDocumentHeightLabel = new javax.swing.JLabel();
236           ivjDocumentHeightLabel.setName("DocumentHeightLabel");
              ivjDocumentHeightLabel.setText("Document␣height:");
238           // user code begin {1}
              // user code end
240        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
```

```
242         // user code end
            handleException(ivjExc);
244      }
         }
246      return ivjDocumentHeightLabel;
      }
248   /**
       * Return the dimensions of a document.
250    * Creation date: (06-09-00 00:46:03)
       * @return java.awt.Dimension
252    */
      public java.awt.Dimension getDocumentSize()
254   {
         int height = getDocumentHeightField().getText().equals("") ?
              getGRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE() :
256                                                Integer.parseInt( getDocumentHeightField().getText() );
         int width = getDocumentWidthField().getText().equals("") ?
              getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() :
258                                                Integer.parseInt( getDocumentWidthField().getText() );
         return new java.awt.Dimension( height, width );
260   }
      /**
262    * Return the DocumentWidthField property value.
       * @return javax.swing.JTextField
264    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
266   private javax.swing.JTextField getDocumentWidthField() {
         if (ivjDocumentWidthField == null) {
268         try {
               ivjDocumentWidthField = new javax.swing.JTextField();
270            ivjDocumentWidthField.setName("DocumentWidthField");
               // user code begin {1}
272            // user code end
            } catch (java.lang.Throwable ivjExc) {
274            // user code begin {2}
               // user code end
276            handleException(ivjExc);
            }
278      }
         return ivjDocumentWidthField;
280   }
      /**
282    * Return the DocumentWidthLabel property value.
       * @return javax.swing.JLabel
284    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
286   private javax.swing.JLabel getDocumentWidthLabel() {
         if (ivjDocumentWidthLabel == null) {
288         try {
               ivjDocumentWidthLabel = new javax.swing.JLabel();
290            ivjDocumentWidthLabel.setName("DocumentWidthLabel");
               ivjDocumentWidthLabel.setText("Document␣width:");
292            // user code begin {1}
               // user code end
294         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
296            // user code end
               handleException(ivjExc);
298         }
         }
300      return ivjDocumentWidthLabel;
      }
302   /**
       * Insert the method's description here.
304    * Creation date: (12-10-00 16:01:25)
       * @return java.lang.String
```

```
306   */
      public final static java.lang.String getGRID_LAYOUT_DOCUMENT_HEIGHT() {
308      return GRID_LAYOUT_DOCUMENT_HEIGHT;
      }
310   /**
       * Insert the method's description here.
312    * Creation date: (12-10-00 16:18:11)
       * @return int
314    */
      public final static int getGRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE() {
316      return GRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE;
      }
318   /**
       * Insert the method's description here.
320    * Creation date: (12-10-00 13:49:28)
       * @return java.lang.String
322    */
      public final static java.lang.String getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() {
324      return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING;
      }
326   /**
       * Insert the method's description here.
328    * Creation date: (12-10-00 14:35:05)
       * @return int
330    */
      public final static int getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE() {
332      return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE;
      }
334   /**
       * Insert the method's description here.
336    * Creation date: (12-10-00 13:49:59)
       * @return java.lang.String
338    */
      public final static java.lang.String getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() {
340      return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING;
      }
342   /**
       * Insert the method's description here.
344    * Creation date: (12-10-00 14:34:25)
       * @return int
346    */
      public final static int getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() {
348      return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE;
      }
350   /**
       * Insert the method's description here.
352    * Creation date: (12-10-00 16:01:47)
       * @return java.lang.String
354    */
      public final static java.lang.String getGRID_LAYOUT_DOCUMENT_WIDTH() {
356      return GRID_LAYOUT_DOCUMENT_WIDTH;
      }
358   /**
       * Insert the method's description here.
360    * Creation date: (12-10-00 16:17:06)
       * @return int
362    */
      public final static int getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() {
364      return GRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE;
      }
366   /**
       * Return the GridLayoutCustomizationBox property value.
368    * @return javax.swing.JDialog
       */
370   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JDialog getGridLayoutCustomizationBox() {
```

```
    if (ivjGridLayoutCustomizationBox == null) {
       try {
          ivjGridLayoutCustomizationBox = new javax.swing.JDialog();
          ivjGridLayoutCustomizationBox.setName("GridLayoutCustomizationBox");
          ivjGridLayoutCustomizationBox.setDefaultCloseOperation(javax.swing.WindowConstants.
              DISPOSE_ON_CLOSE);
          ivjGridLayoutCustomizationBox.setBounds(24, 44, 398, 300);
          ivjGridLayoutCustomizationBox.setModal(true);
          ivjGridLayoutCustomizationBox.setTitle("Grid␣layout␣customization");
          getGridLayoutCustomizationBox().setContentPane(getJDialogContentPane());
          // user code begin {1}
          // user code end
       } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
          // user code end
          handleException(ivjExc);
       }
    }
    return ivjGridLayoutCustomizationBox;
 }
 /**
  * Return the HorizontalSpacingField property value.
  * @return javax.swing.JTextField
  */
 /* WARNING: THIS METHOD WILL BE REGENERATED. */
 private javax.swing.JTextField getHorizontalSpacingField() {
    if (ivjHorizontalSpacingField == null) {
       try {
          ivjHorizontalSpacingField = new javax.swing.JTextField();
          ivjHorizontalSpacingField.setName("HorizontalSpacingField");
          // user code begin {1}
          // user code end
       } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
          // user code end
          handleException(ivjExc);
       }
    }
    return ivjHorizontalSpacingField;
 }
 /**
  * Return the HorizontalSpacingLabel property value.
  * @return javax.swing.JLabel
  */
 /* WARNING: THIS METHOD WILL BE REGENERATED. */
 private javax.swing.JLabel getHorizontalSpacingLabel() {
    if (ivjHorizontalSpacingLabel == null) {
       try {
          ivjHorizontalSpacingLabel = new javax.swing.JLabel();
          ivjHorizontalSpacingLabel.setName("HorizontalSpacingLabel");
          ivjHorizontalSpacingLabel.setText("Horizontal␣spacing:");
          // user code begin {1}
          // user code end
       } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
          // user code end
          handleException(ivjExc);
       }
    }
    return ivjHorizontalSpacingLabel;
 }
 /**
  * Return the JDialogContentPane property value.
  * @return javax.swing.JPanel
  */
 /* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
    private javax.swing.JPanel getJDialogContentPane() {
438    if (ivjJDialogContentPane == null) {
          try {
440          ivjJDialogContentPane = new javax.swing.JPanel();
             ivjJDialogContentPane.setName("JDialogContentPane");
442          ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
             getJDialogContentPane().add(getMainPanel(), "Center");
444          getJDialogContentPane().add(getButtonPanel(), "South");
             // user code begin {1}
446          // user code end
          } catch (java.lang.Throwable ivjExc) {
448          // user code begin {2}
             // user code end
450          handleException(ivjExc);
          }
452    }
       return ivjJDialogContentPane;
454 }
    /**
456  * Return the MainPanel property value.
     * @return javax.swing.JPanel
458  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
460 private javax.swing.JPanel getMainPanel() {
       if (ivjMainPanel == null) {
462       try {
             ivjMainPanel = new javax.swing.JPanel();
464          ivjMainPanel.setName("MainPanel");
             ivjMainPanel.setLayout(new java.awt.GridBagLayout());
466
             java.awt.GridBagConstraints constraintsDocumentWidthLabel = new java.awt.GridBagConstraints();
468          constraintsDocumentWidthLabel.gridx = 0; constraintsDocumentWidthLabel.gridy = 0;
             constraintsDocumentWidthLabel.anchor = java.awt.GridBagConstraints.WEST;
470          constraintsDocumentWidthLabel.weighty = 1.0;
             constraintsDocumentWidthLabel.insets = new java.awt.Insets(4, 4, 4, 4);
472          getMainPanel().add(getDocumentWidthLabel(), constraintsDocumentWidthLabel);

474          java.awt.GridBagConstraints constraintsDocumentHeightLabel = new java.awt.GridBagConstraints();
             constraintsDocumentHeightLabel.gridx = 0; constraintsDocumentHeightLabel.gridy = 1;
476          constraintsDocumentHeightLabel.anchor = java.awt.GridBagConstraints.WEST;
             constraintsDocumentHeightLabel.weighty = 1.0;
478          constraintsDocumentHeightLabel.insets = new java.awt.Insets(4, 4, 4, 4);
             getMainPanel().add(getDocumentHeightLabel(), constraintsDocumentHeightLabel);
480
             java.awt.GridBagConstraints constraintsHorizontalSpacingLabel = new java.awt.GridBagConstraints
                 ();
482          constraintsHorizontalSpacingLabel.gridx = 0; constraintsHorizontalSpacingLabel.gridy = 2;
             constraintsHorizontalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
484          constraintsHorizontalSpacingLabel.weighty = 1.0;
             constraintsHorizontalSpacingLabel.insets = new java.awt.Insets(4, 4, 4, 4);
486          getMainPanel().add(getHorizontalSpacingLabel(), constraintsHorizontalSpacingLabel);

488          java.awt.GridBagConstraints constraintsVerticalSpacingLabel = new java.awt.GridBagConstraints();
             constraintsVerticalSpacingLabel.gridx = 0; constraintsVerticalSpacingLabel.gridy = 3;
490          constraintsVerticalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
             constraintsVerticalSpacingLabel.weighty = 1.0;
492          constraintsVerticalSpacingLabel.insets = new java.awt.Insets(4, 4, 4, 4);
             getMainPanel().add(getVerticalSpacingLabel(), constraintsVerticalSpacingLabel);
494
             java.awt.GridBagConstraints constraintsDocumentWidthField = new java.awt.GridBagConstraints();
496          constraintsDocumentWidthField.gridx = 1; constraintsDocumentWidthField.gridy = 0;
             constraintsDocumentWidthField.fill = java.awt.GridBagConstraints.HORIZONTAL;
498          constraintsDocumentWidthField.anchor = java.awt.GridBagConstraints.EAST;
             constraintsDocumentWidthField.weightx = 1.0;
500          constraintsDocumentWidthField.weighty = 1.0;
             constraintsDocumentWidthField.insets = new java.awt.Insets(4, 4, 4, 4);
```

```
502         getMainPanel().add(getDocumentWidthField(), constraintsDocumentWidthField);

504         java.awt.GridBagConstraints constraintsDocumentHeightField = new java.awt.GridBagConstraints();
            constraintsDocumentHeightField.gridx = 1; constraintsDocumentHeightField.gridy = 1;
506         constraintsDocumentHeightField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsDocumentHeightField.anchor = java.awt.GridBagConstraints.EAST;
508         constraintsDocumentHeightField.weightx = 1.0;
            constraintsDocumentHeightField.weighty = 1.0;
510         constraintsDocumentHeightField.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getDocumentHeightField(), constraintsDocumentHeightField);
512
            java.awt.GridBagConstraints constraintsHorizontalSpacingField = new java.awt.GridBagConstraints
                 ();
514         constraintsHorizontalSpacingField.gridx = 1; constraintsHorizontalSpacingField.gridy = 2;
            constraintsHorizontalSpacingField.fill = java.awt.GridBagConstraints.HORIZONTAL;
516         constraintsHorizontalSpacingField.anchor = java.awt.GridBagConstraints.EAST;
            constraintsHorizontalSpacingField.weightx = 1.0;
518         constraintsHorizontalSpacingField.weighty = 1.0;
            constraintsHorizontalSpacingField.insets = new java.awt.Insets(4, 4, 4, 4);
520         getMainPanel().add(getHorizontalSpacingField(), constraintsHorizontalSpacingField);

522         java.awt.GridBagConstraints constraintsVerticalSpacingField = new java.awt.GridBagConstraints();
            constraintsVerticalSpacingField.gridx = 1; constraintsVerticalSpacingField.gridy = 3;
524         constraintsVerticalSpacingField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsVerticalSpacingField.anchor = java.awt.GridBagConstraints.EAST;
526         constraintsVerticalSpacingField.weightx = 1.0;
            constraintsVerticalSpacingField.weighty = 1.0;
528         constraintsVerticalSpacingField.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getVerticalSpacingField(), constraintsVerticalSpacingField);
530         // user code begin {1}
            ivjMainPanel.setBorder( ClientConstants.createDialogBorder( ivjMainPanel.getBackground() ));
532         // user code end
         } catch (java.lang.Throwable ivjExc) {
534         // user code begin {2}
            // user code end
536         handleException(ivjExc);
         }
538   }
      return ivjMainPanel;
540 }
    /**
542  * Return the OKButton property value.
     * @return javax.swing.JButton
544  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
546 private javax.swing.JButton getOKButton() {
      if (ivjOKButton == null) {
548      try {
            ivjOKButton = new javax.swing.JButton();
550         ivjOKButton.setName("OKButton");
            ivjOKButton.setMnemonic('O');
552         ivjOKButton.setText("OK");
            // user code begin {1}
554         ivjOKButton.addActionListener( new OKHandler() );
            // user code end
556      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
558         // user code end
            handleException(ivjExc);
560      }
      }
562   return ivjOKButton;
    }
564 /**
     * Return the VerticalSpacingField property value.
566  * @return javax.swing.JTextField
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextField getVerticalSpacingField() {
   if (ivjVerticalSpacingField == null) {
      try {
         ivjVerticalSpacingField = new javax.swing.JTextField();
         ivjVerticalSpacingField.setName("VerticalSpacingField");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjVerticalSpacingField;
}
/**
 * Return the VerticalSpacingLabel property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getVerticalSpacingLabel() {
   if (ivjVerticalSpacingLabel == null) {
      try {
         ivjVerticalSpacingLabel = new javax.swing.JLabel();
         ivjVerticalSpacingLabel.setName("VerticalSpacingLabel");
         ivjVerticalSpacingLabel.setText("Vertical␣spacing:");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjVerticalSpacingLabel;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   }
   // user code begin {2}
   // user code end
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 15:31:05)
 * @return boolean
```

```
 */
public boolean isCommitted() {
   return committed;
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      GridPanoramaLayoutManager aGridPanoramaLayoutManager;
      aGridPanoramaLayoutManager = new GridPanoramaLayoutManager();
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣clientapp.ClientPanoramaLayout");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (12-10-00 15:31:05)
 * @param newCommitted boolean
 */
public void setCommitted(boolean newCommitted) {
   committed = newCommitted;
}
/**
 * Set document specific preferences.
 * Creation date: (12-10-00 13:46:17)
 * @param newSettings java.util.Properties
 */
public void setSettings(java.util.Properties newSettings)
{
   if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() )!= null )
   {
      getDocumentHeightField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() ));
   }
   else
   {
      getDocumentHeightField().setText( String.valueOf( getDOCUMENT_HEIGHT_DEFAULT_VALUE() ));
   }
   if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() )!= null )
   {
      getDocumentWidthField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() ));
   }
   else
   {
      getDocumentWidthField().setText( String.valueOf( getDOCUMENT_WIDTH_DEFAULT_VALUE() ));
   }
   if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() )!= null )
   {
      getHorizontalSpacingField().setText( newSettings.getProperty(
           getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() ));
   }
   else
   {
      getHorizontalSpacingField().setText( String.valueOf(
           getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ));
   }
   if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() )!= null )
   {
      getVerticalSpacingField().setText( newSettings.getProperty(
           getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() ));
   }
   else
   {
```

```
        getVerticalSpacingField().setText( String.valueOf(
            getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ));
696     }
    }
698 /**
     * Insert the method's description here.
700  * Creation date: (12-10-00 15:37:30)
     * @param owner javax.swing.JDialog
702  */
    public void show(javax.swing.JDialog owner)
704 {
        getGridLayoutCustomizationBox().setLocationRelativeTo( owner );
706     getGridLayoutCustomizationBox().show();
    }
708 /**
     * Take a properties table and update or insert (if non-existent) the <key,value> pairs defining the
         configuration of this layout manager
710  * Creation date: (12-10-00 14:49:58)
     * @return java.util.Properties
712  * @param currentSettings java.util.Properties
     */
714 public java.util.Properties updateSettings(java.util.Properties currentSettings)
    {
716     currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT(), getDocumentHeightField().getText() );
        currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_WIDTH(), getDocumentWidthField().getText() );
718     currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING(), getHorizontalSpacingField
            ().getText() );
        currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING(), getVerticalSpacingField().
            getText() );
720     return currentSettings;
    }
722 /**
     * Validate the contents of the document preferences configuration box.
724  * Creation date: (06-09-00 13:55:19)
     * @return boolean
726  */
    public boolean validateFields()
728 {
        peerviewmisc.MessageBox messageBox = new peerviewmisc.MessageBox();
730     boolean valid = true;

732     if (  ( Integer.parseInt( getDocumentHeightField().getText() )> ClientConstants.
            getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() )
            || ( Integer.parseInt( getDocumentHeightField().getText() )< ClientConstants.
                getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() ))
734     {
            messageBox.setText( ClientConstants.getDOCUMENT_SIZE_OUT_OF_BOUNDS() );
736         valid = false;
        }
738     else if (  ( Integer.parseInt( getDocumentWidthField().getText() )> ClientConstants.
            getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() )
            ||    ( Integer.parseInt( getDocumentWidthField().getText() )< ClientConstants.
                getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() ))
740
        {
742         messageBox.setText( ClientConstants.getDOCUMENT_SIZE_OUT_OF_BOUNDS() );
            valid = false;
744     }
        else if (  ( Integer.parseInt( getHorizontalSpacingField().getText() )> ClientConstants.
            getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() )
746         || ( Integer.parseInt( getHorizontalSpacingField().getText() )< ClientConstants.
                getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() ))
        {
748         messageBox.setText( ClientConstants.getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() );
            valid = false;
750     }
```

```
      else if (  ( Integer.parseInt( getVerticalSpacingField().getText() )> ClientConstants.
          getMAX_GRID_LAYOUT_VERTICAL_SPACING() )
752      ||    ( Integer.parseInt( getVerticalSpacingField().getText() )< ClientConstants.
             getMIN_GRID_LAYOUT_VERTICAL_SPACING() ))
      {
754      messageBox.setText( ClientConstants.getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() );
         valid = false;
756   }
      if ( valid == false )
758   {
         messageBox.show();
760   }
      return valid;
762 }
    }
```

Listing C.13: GroupDirectory.java

```
    package clientapp;
 2
    /**
 4   * The client group directory where the listing of available along with their properties such as number
         of participants
     * and data volume are displayed.
 6   * Creation date: (30-06-00 20:37:33)
     * @author:
 8   */
     import peerviewmisc.*;
10   import java.awt.event.*;
     import com.sun.media.jsdt.*;
12   import java.util.Collection;
     import java.util.Iterator;
14   import java.util.Vector;
     import java.util.Enumeration;
16
    public class GroupDirectory extends javax.swing.JDialog {
18
       public class GroupDirectoryWindowListener extends WindowAdapter
20     {
          public void windowActivated( WindowEvent we )
22        {
             initWindow();
24        }
       }
26
       public class CreateGroupButtonHandler implements ActionListener
28     {
          public void actionPerformed ( ActionEvent ae )
30        {
             createNewGroup();
32        }
       }
34
       public class EditGroupButtonHandler implements ActionListener
36     {
          public void actionPerformed ( ActionEvent ae )
38        {
             // System.out.println( "ramte editexistinggroup" );
40           editExistingGroup();
          }
42     }

44     public class DeleteGroupButtonHandler implements ActionListener
       {
46        public void actionPerformed ( ActionEvent ae )
          {
48           // System.out.println( "ramte deletegroupbuttonhandler" );
```

```
            deleteGroup();
50       }
      }
52
      public class JoinGroupButtonHandler implements ActionListener
54    {
         public void actionPerformed ( ActionEvent ae )
56       {
            joinGroup();
58       }
      }
60
      public class CloseButtonHandler implements ActionListener
62    {
         public void actionPerformed( ActionEvent ae )
64       {
            dispose();
66       }
      }
68    private javax.swing.JButton ivjCloseButton = null;
      private javax.swing.JButton ivjCreateGroupButton = null;
70    private javax.swing.JButton ivjDeleteGroupButton = null;
      private javax.swing.JPanel ivjJDialogContentPane = null;
72    private javax.swing.JButton ivjJoinGroupButton = null;
      private javax.swing.JPanel ivjJPanel2 = null;
74    private java.awt.FlowLayout ivjJPanel2FlowLayout = null;
      private javax.swing.JPanel ivjJPanel3 = null;
76    private java.awt.FlowLayout ivjJPanel3FlowLayout = null;
      private javax.swing.JScrollPane ivjJScrollPane1 = null;
78    private javax.swing.JTable ivjScrollPaneTable = null;
      private DeleteGroup ivjDeleteGroup1 = null;
80    private NewGroup ivjNewGroup1 = null;
      private peerviewmisc.WorkGroup workGroup = null;
82    private EditGroup ivjEditGroup1 = null;
      private javax.swing.JButton ivjEditGroupButton = null;
84    private javax.swing.table.DefaultTableModel tableModel;
      private javax.swing.JPanel ivjButtonPanel = null;
86    private javax.swing.JPanel ivjTablePanel = null;
      private peerviewmisc.WorkGroup[] groupDirectoryValues = null;
88    IvjEventHandler ivjEventHandler = new IvjEventHandler();
      private ClientApplication clientApplication = null;
90
   class IvjEventHandler implements java.awt.event.ActionListener {
92       public void actionPerformed(java.awt.event.ActionEvent e) {
            if (e.getSource() == GroupDirectory.this.getJoinGroupButton())
94             connEtoM1(e);
            if (e.getSource() == GroupDirectory.this.getCreateGroupButton())
96             connEtoM2(e);
            if (e.getSource() == GroupDirectory.this.getEditGroupButton())
98             connEtoM3(e);
            if (e.getSource() == GroupDirectory.this.getDeleteGroupButton())
100            connEtoM4(e);
            if (e.getSource() == GroupDirectory.this.getCloseButton())
102            connEtoM5(e);
         };
104   };
   /**
106 * GroupDirectory constructor comment.
    */
108 public GroupDirectory() {
      super();
110   initialize();
   }
112 /**
    * GroupDirectory constructor comment.
114 * @param owner java.awt.Dialog
```

```
     */
116  public GroupDirectory(java.awt.Dialog owner) {
        super(owner);
118  }
     /**
120   * GroupDirectory constructor comment.
      * @param owner java.awt.Dialog
122   * @param title java.lang.String
      */
124  public GroupDirectory(java.awt.Dialog owner, String title) {
        super(owner, title);
126  }
     /**
128   * GroupDirectory constructor comment.
      * @param owner java.awt.Dialog
130   * @param title java.lang.String
      * @param modal boolean
132   */
     public GroupDirectory(java.awt.Dialog owner, String title, boolean modal) {
134     super(owner, title, modal);
     }
136  /**
      * GroupDirectory constructor comment.
138   * @param owner java.awt.Dialog
      * @param modal boolean
140   */
     public GroupDirectory(java.awt.Dialog owner, boolean modal) {
142     super(owner, modal);
     }
144  /**
      * GroupDirectory constructor comment.
146   * @param owner java.awt.Frame
      */
148  public GroupDirectory(java.awt.Frame owner) {
        super(owner);
150  }
     /**
152   * GroupDirectory constructor comment.
      * @param owner java.awt.Frame
154   * @param title java.lang.String
      */
156  public GroupDirectory(java.awt.Frame owner, String title) {
        super(owner, title);
158  }
     /**
160   * GroupDirectory constructor comment.
      * @param owner java.awt.Frame
162   * @param title java.lang.String
      * @param modal boolean
164   */
     public GroupDirectory(java.awt.Frame owner, String title, boolean modal) {
166     super(owner, title, modal);
     }
168  /**
      * GroupDirectory constructor comment.
170   * @param owner java.awt.Frame
      * @param modal boolean
172   */
     public GroupDirectory(java.awt.Frame owner, boolean modal) {
174     super(owner, modal);
     }
176  /**
      * connEtoM1: (JoinGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
          joinGroup()V)
178   * @param arg1 java.awt.event.ActionEvent
      */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM1(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.joinGroup();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM2: (CreateGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
     createNewGroup()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM2(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.createNewGroup();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM3: (EditGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
     editExistingGroup()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM3(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // System.out.println( "FYR" );
      // user code end
      this.editExistingGroup();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM4: (DeleteGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
     deleteGroup()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM4(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.deleteGroup();
      // user code begin {2}
      // user code end
```

```
      } catch (java.lang.Throwable ivjExc) {
244      // user code begin {3}
         // user code end
246      handleException(ivjExc);
      }
248 }
    /**
250  * connEtoM5: (CloseButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.dispose
         ()V)
     * @param arg1 java.awt.event.ActionEvent
252  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
254 private void connEtoM5(java.awt.event.ActionEvent arg1) {
      try {
256      // user code begin {1}
         // user code end
258      this.dispose();
         // user code begin {2}
260      // user code end
      } catch (java.lang.Throwable ivjExc) {
262      // user code begin {3}
         // user code end
264      handleException(ivjExc);
      }
266 }
    /**
268  * Display a dialog box for entering information about a new group and create a corresponding object if
         the user acknowledges
     * his or her entries.
270  * Creation date: (03-07-00 21:44:35)
     */
272 public void createNewGroup()
    {
274   NewGroup newGroup = getNewGroup1();

276   newGroup.setWorkGroup( new WorkGroup() );
      newGroup.show();
278   WorkGroup workGroup = newGroup.getWorkGroup();
      // if the workgroup box wasn't cancelled
280   if ( newGroup.isIsDiscarded() == false )
      {
282      // complete the non-editable fields of the new workgroup
         workGroup.setCreationDate( new java.util.Date() );
284      workGroup.setCreator( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
             getAUTHOR_NAME() ));
         getClientApplication().getClientManager().createWorkGroup( workGroup );
286   }
    }
288 /**
     * Delete whatever group is currently selected in the group directory listing if the user acknowledges
         the deletion.
290  * Creation date: (03-07-00 21:45:09)
     */
292 public void deleteGroup()
    {
294   int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
      // if no selection
296   if ( selectedRow == -1 )
      {
298      MessageBox messageBox = new MessageBox();
         messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_DELETE() );
300      messageBox.show();
      }
302   else
      {  WorkGroup selectedGroup = (WorkGroup) groupDirectoryValues[ selectedRow ];
304      // if the selected group has active participants
```

```
        if ( selectedGroup.getParticipants().size() > 0 )
306     {
           MessageBox messageBox = new MessageBox();
308        messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS());
           messageBox.show();
310        return;
        }
312     ConfirmationBox confirmationBox = new ConfirmationBox();
        confirmationBox.setText( ClientConstants.getCONFIRM_WORKGROUP_DELETION( selectedGroup.getGroupName
             () ));
314     confirmationBox.show();
        if ( confirmationBox.isConfirmed() == true )
316     {
           getClientApplication().getClientManager().removeWorkGroup( selectedGroup );
318     }
     }
320  }
     /**
322   * Disable all buttons that trigger actions involving communication with the server. Used in case of an
         inactive server
      * or a broken communications channel.
324   * Creation date: (10-08-00 16:13:10)
      */
326  private void disableButtons()
     {
328     getJoinGroupButton().setEnabled( false);
        getCreateGroupButton().setEnabled( false );
330     getEditGroupButton().setEnabled( false );
        getDeleteGroupButton().setEnabled( false );
332  }
     /**
334   * Display an edit box containing the attributes of the group currently selected in the group directory.
      * Creation date: (03-07-00 21:44:52)
336   */
     public void editExistingGroup()
338  {
        int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
340     // if no selection
        if ( selectedRow == -1 )
342     {
           MessageBox messageBox = new MessageBox();
344        messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() );
           messageBox.show();
346     }
        else
348     {
           WorkGroup backup = new WorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
350
           // if the selected group has active participants
352        if ( backup.getParticipants().size() > 0 )
           {
354           MessageBox messageBox = new MessageBox();
              messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() );
356           messageBox.show();
              return;
358        }
           getEditGroup1().setWorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
360        getEditGroup1().show();
           if ( getEditGroup1().isIsDiscarded() == false )
362        {
              getClientApplication().getClientManager().removeWorkGroup( backup );
364           WorkGroup newWorkGroup = getEditGroup1().getWorkGroup();
              newWorkGroup.setCreationDate( new java.util.Date() );
366           newWorkGroup.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.getAUTHOR_NAME
                 () ));
```

```
            // the below automatically causes the server to broadcast a group directory update to all
                clients, thus
368         // eliminating the need to update the locally cached copy in the clientInfo object.
            getClientApplication().getClientManager().createWorkGroup( newWorkGroup );
370       }
        }
372  }
     /**
374   * Insert the method's description here.
      * Creation date: (10-08-00 16:13:27)
376   */
     private void enableButtons()
378  {
        getJoinGroupButton().setEnabled( true );
380     getCreateGroupButton().setEnabled( true );
        getEditGroupButton().setEnabled( true );
382     getDeleteGroupButton().setEnabled( true );

384  }
     /**
386   * Return the JPanel1 property value.
      * @return javax.swing.JPanel
388   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
390  private javax.swing.JPanel getButtonPanel() {
        if (ivjButtonPanel == null) {
392        try {
            ivjButtonPanel = new javax.swing.JPanel();
394         ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(new java.awt.BorderLayout());
396         getButtonPanel().add(getJPanel3(), "East");
            getButtonPanel().add(getJPanel2(), "West");
398         // user code begin {1}
            getButtonPanel().setBorder( javax.swing.BorderFactory.createMatteBorder(
400                                 ClientConstants.getBORDER_WIDTH(),
                                    ClientConstants.getBORDER_WIDTH(),
402                                 ClientConstants.getBORDER_WIDTH(),
                                    ClientConstants.getBORDER_WIDTH(),
404                                 getButtonPanel().getBackground() ));
            // user code end
406        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
408         // user code end
            handleException(ivjExc);
410        }
        }
412     return ivjButtonPanel;
     }
414  /**
      * Insert the method's description here.
416   * Creation date: (10-08-00 15:41:17)
      * @return clientapp.ClientApplication
418   */
     public ClientApplication getClientApplication() {
420     return clientApplication;
     }
422  /**
      * Return the CloseButton property value.
424   * @return javax.swing.JButton
      */
426  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getCloseButton() {
428     if (ivjCloseButton == null) {
           try {
430         ivjCloseButton = new javax.swing.JButton();
            ivjCloseButton.setName("CloseButton");
```

```
432         ivjCloseButton.setMnemonic('c');
            ivjCloseButton.setText("Close");
434         // user code begin {1}
            ivjCloseButton.addActionListener( new CloseButtonHandler() );
436         // user code end
         } catch (java.lang.Throwable ivjExc) {
438         // user code begin {2}
            // user code end
440         handleException(ivjExc);
         }
442      }
         return ivjCloseButton;
444   }
      /**
446    * Return the CreateGroupButton property value.
       * @return javax.swing.JButton
448    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
450   private javax.swing.JButton getCreateGroupButton() {
         if (ivjCreateGroupButton == null) {
452         try {
            ivjCreateGroupButton = new javax.swing.JButton();
454         ivjCreateGroupButton.setName("CreateGroupButton");
            ivjCreateGroupButton.setMnemonic('g');
456         ivjCreateGroupButton.setText("Create␣group");
            // user code begin {1}
458         //ivjCreateGroupButton.addActionListener( new CreateGroupButtonHandler() );
            // user code end
460         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
462         // user code end
            handleException(ivjExc);
464         }
         }
466      return ivjCreateGroupButton;
      }
468   /**
       * Return the DeleteGroup1 property value.
470    * @return peerviewmisc.DeleteGroup
       */
472   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      public peerviewmisc.DeleteGroup getDeleteGroup1() {
474      if (ivjDeleteGroup1 == null) {
         try {
476         ivjDeleteGroup1 = new peerviewmisc.DeleteGroup();
            ivjDeleteGroup1.setName("DeleteGroup1");
478         // user code begin {1}
            // user code end
480         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
482         // user code end
            handleException(ivjExc);
484         }
         }
486      return ivjDeleteGroup1;
      }
488   /**
       * Return the DeleteGroupButton property value.
490    * @return javax.swing.JButton
       */
492   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getDeleteGroupButton() {
494      if (ivjDeleteGroupButton == null) {
         try {
496         ivjDeleteGroupButton = new javax.swing.JButton();
            ivjDeleteGroupButton.setName("DeleteGroupButton");
```

```
498         ivjDeleteGroupButton.setMnemonic('D');
            ivjDeleteGroupButton.setText("Delete␣group");
500         // user code begin {1}
            //ivjDeleteGroupButton.addActionListener( new DeleteGroupButtonHandler() );
502         // user code end
         } catch (java.lang.Throwable ivjExc) {
504         // user code begin {2}
            // user code end
506         handleException(ivjExc);
         }
508      }
         return ivjDeleteGroupButton;
510   }
      /**
512    * Return the EditGroup1 property value.
       * @return peerviewmisc.EditGroup
514    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
516   public peerviewmisc.EditGroup getEditGroup1() {
         if (ivjEditGroup1 == null) {
518         try {
            ivjEditGroup1 = new peerviewmisc.EditGroup();
520         ivjEditGroup1.setName("EditGroup1");
            ivjEditGroup1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
522         // user code begin {1}
            ivjEditGroup1.setGroupDirectory( this );
524         // user code end
         } catch (java.lang.Throwable ivjExc) {
526         // user code begin {2}
            // user code end
528         handleException(ivjExc);
         }
530      }
         return ivjEditGroup1;
532   }
      /**
534    * Return the EditGroupButton property value.
       * @return javax.swing.JButton
536    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
538   public javax.swing.JButton getEditGroupButton() {
         if (ivjEditGroupButton == null) {
540         try {
            ivjEditGroupButton = new javax.swing.JButton();
542         ivjEditGroupButton.setName("EditGroupButton");
            ivjEditGroupButton.setMnemonic('E');
544         ivjEditGroupButton.setText("Edit␣group");
            // user code begin {1}
546         //ivjEditGroupButton.addActionListener( new EditGroupButtonHandler() );
            // user code end
548         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
550         // user code end
            handleException(ivjExc);
552         }
         }
554      return ivjEditGroupButton;
      }
556   /**
       * Insert the method's description here.
558    * Creation date: (26-07-00 21:09:59)
       * @return peerviewmisc.WorkGroup[]
560    */
      public peerviewmisc.WorkGroup[] getGroupDirectoryValues() {
562      return groupDirectoryValues;
      }
```

```
/**
 * Return the JDialogContentPane property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJDialogContentPane() {
   if (ivjJDialogContentPane == null) {
      try {
         ivjJDialogContentPane = new javax.swing.JPanel();
         ivjJDialogContentPane.setName("JDialogContentPane");
         ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
         getJDialogContentPane().add(getTablePanel(), "Center");
         getJDialogContentPane().add(getButtonPanel(), "South");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjJDialogContentPane;
}
/**
 * Return the JoinGroupButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getJoinGroupButton() {
   if (ivjJoinGroupButton == null) {
      try {
         ivjJoinGroupButton = new javax.swing.JButton();
         ivjJoinGroupButton.setName("JoinGroupButton");
         ivjJoinGroupButton.setMnemonic('J');
         ivjJoinGroupButton.setText("Join␣group");
         // user code begin {1}
         //ivjJoinGroupButton.addActionListener( new JoinGroupButtonHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjJoinGroupButton;
}
/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel2() {
   if (ivjJPanel2 == null) {
      try {
         ivjJPanel2 = new javax.swing.JPanel();
         ivjJPanel2.setName("JPanel2");
         ivjJPanel2.setLayout(getJPanel2FlowLayout());
         getJPanel2().add(getJoinGroupButton(), getJoinGroupButton().getName());
         getJPanel2().add(getCreateGroupButton(), getCreateGroupButton().getName());
         ivjJPanel2.add(getEditGroupButton());
         ivjJPanel2.add(getDeleteGroupButton());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
```

```
630         handleException(ivjExc);
         }
632     }
        return ivjJPanel2;
634  }
     /**
636   * Return the JPanel2FlowLayout property value.
      * @return java.awt.FlowLayout
638   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
640  private java.awt.FlowLayout getJPanel2FlowLayout() {
        java.awt.FlowLayout ivjJPanel2FlowLayout = null;
642     try {
           /* Create part */
644        ivjJPanel2FlowLayout = new java.awt.FlowLayout();
           ivjJPanel2FlowLayout.setAlignment(java.awt.FlowLayout.LEFT);
646     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
648     };
        return ivjJPanel2FlowLayout;
650  }
     /**
652   * Return the JPanel3 property value.
      * @return javax.swing.JPanel
654   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
656  private javax.swing.JPanel getJPanel3() {
        if (ivjJPanel3 == null) {
658        try {
              ivjJPanel3 = new javax.swing.JPanel();
660           ivjJPanel3.setName("JPanel3");
              ivjJPanel3.setLayout(getJPanel3FlowLayout());
662           getJPanel3().add(getCloseButton(), getCloseButton().getName());
              // user code begin {1}
664           // user code end
           } catch (java.lang.Throwable ivjExc) {
666           // user code begin {2}
              // user code end
668           handleException(ivjExc);
           }
670     }
        return ivjJPanel3;
672  }
     /**
674   * Return the JPanel3FlowLayout property value.
      * @return java.awt.FlowLayout
676   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
678  private java.awt.FlowLayout getJPanel3FlowLayout() {
        java.awt.FlowLayout ivjJPanel3FlowLayout = null;
680     try {
           /* Create part */
682        ivjJPanel3FlowLayout = new java.awt.FlowLayout();
           ivjJPanel3FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
684     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
686     };
        return ivjJPanel3FlowLayout;
688  }
     /**
690   * Return the JScrollPane1 property value.
      * @return javax.swing.JScrollPane
692   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
694  private javax.swing.JScrollPane getJScrollPane1() {
        if (ivjJScrollPane1 == null) {
```

```
696         try {
               ivjJScrollPane1 = new javax.swing.JScrollPane();
698            ivjJScrollPane1.setName("JScrollPane1");
               ivjJScrollPane1.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
700            ivjJScrollPane1.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
                   );
               ivjJScrollPane1.setDoubleBuffered(true);
702            getJScrollPane1().setViewportView(getScrollPaneTable());
               // user code begin {1}
704            // user code end
            } catch (java.lang.Throwable ivjExc) {
706            // user code begin {2}
               // user code end
708            handleException(ivjExc);
            }
710      }
         return ivjJScrollPane1;
712   }
      /**
714    * Return the NewGroup1 property value.
       * @return peerviewmisc.NewGroup
716    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
718   public peerviewmisc.NewGroup getNewGroup1() {
         if (ivjNewGroup1 == null) {
720         try {
               ivjNewGroup1 = new peerviewmisc.NewGroup();
722            ivjNewGroup1.setName("NewGroup1");
               // user code begin {1}
724            ivjNewGroup1.setGroupDirectory( this );
               // user code end
726         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
728            // user code end
               handleException(ivjExc);
730         }
         }
732      return ivjNewGroup1;
      }
734   /**
       * Return the ScrollPaneTable property value.
736    * @return javax.swing.JTable
       */
738   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTable getScrollPaneTable() {
740      if (ivjScrollPaneTable == null) {
            try {
742            ivjScrollPaneTable = new javax.swing.JTable();
               ivjScrollPaneTable.setName("ScrollPaneTable");
744            getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
               getJScrollPane1().getViewport().setBackingStoreEnabled(true);
746            ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
               ivjScrollPaneTable.setPreferredSize(new java.awt.Dimension(716,412));
748            ivjScrollPaneTable.setBounds(0, 0, 716, 412);
               ivjScrollPaneTable.setPreferredScrollableViewportSize(new java.awt.Dimension(700, 400));
750            ivjScrollPaneTable.setAutoCreateColumnsFromModel(true);
               // user code begin {1}
752            ivjScrollPaneTable.setSelectionMode( javax.swing.ListSelectionModel.SINGLE_SELECTION );
               ivjScrollPaneTable.setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
754            // user code end
            } catch (java.lang.Throwable ivjExc) {
756            // user code begin {2}
               // user code end
758            handleException(ivjExc);
            }
760      }
```

```
      return ivjScrollPaneTable;
   }
   /**
    * Insert the method's description here.
    * Creation date: (04-07-00 19:38:32)
    * @return javax.swing.table.DefaultTableModel
    */
   public javax.swing.table.DefaultTableModel getTableModel() {
      return tableModel;
   }
   /**
    * Return the JPanel4 property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getTablePanel() {
      if (ivjTablePanel == null) {
         try {
            ivjTablePanel = new javax.swing.JPanel();
            ivjTablePanel.setName("TablePanel");
            ivjTablePanel.setLayout(new java.awt.BorderLayout());
            getTablePanel().add(getJScrollPane1(), "Center");
            // user code begin {1}
            getTablePanel().setBorder( ClientConstants.createDialogBorder( getTablePanel().getBackground() )
                 );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjTablePanel;
   }
   /**
    * Insert the method's description here.
    * Creation date: (03-07-00 21:00:14)
    * @return peerviewmisc.WorkGroup
    */
   public peerviewmisc.WorkGroup getWorkGroup() {
      return workGroup;
   }
   /**
    * Traverses the group directory (list of active groups) to determine if any group matches the group
        name and description parameters
    * Creation date: (26-07-00 16:59:38)
    * @return boolean
    * @param groupName java.lang.String
    * @param description java.lang.String
    */
   public boolean groupExists(String groupName, String description)
   {
      boolean exists = false;
      for ( int i = 0; i < groupDirectoryValues.length; i++ )
      {
         if ( ( (WorkGroup) groupDirectoryValues[i]).getGroupName().equalsIgnoreCase( groupName )&&
             ( (WorkGroup) groupDirectoryValues[i]).getDescription().equalsIgnoreCase( description ))
         {
            exists = true;
            break;
         }
      }
      return exists;
   }
   /**
    * Called whenever the part throws an exception.
```

```
     * @param exception java.lang.Throwable
826   */
     private void handleException(java.lang.Throwable exception) {
828
        /* Uncomment the following lines to print uncaught exceptions to stdout */
830     // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
832  }
     /**
834   * Initializes connections
     * @exception java.lang.Exception The exception description.
836   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
838  private void initConnections() throws java.lang.Exception {
        // user code begin {1}
840     // user code end
        getJoinGroupButton().addActionListener(ivjEventHandler);
842     getCreateGroupButton().addActionListener(ivjEventHandler);
        getEditGroupButton().addActionListener(ivjEventHandler);
844     getDeleteGroupButton().addActionListener(ivjEventHandler);
        getCloseButton().addActionListener(ivjEventHandler);
846  }
     /**
848   * Initialize the class.
     */
850  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
852     try {
           // user code begin {1}
854        // user code end
           setName("GroupDirectory");
856        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setSize(845, 467);
858        setModal(true);
           setTitle("Group␣directory");
860        setContentPane(getJDialogContentPane());
           initConnections();
862     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
864     }
        // user code begin {2}
866     getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS );
        setLocationRelativeTo( getOwner() );
868     addWindowListener( new GroupDirectoryWindowListener() );
        // user code end
870  }
     /**
872   * Insert the method's description here.
     * Creation date: (04-07-00 23:11:34)
874   */
     public void initTable()
876  {
        WorkGroup workGroup = new WorkGroup();
878     tableModel = new javax.swing.table.DefaultTableModel();
        ivjScrollPaneTable.setModel( tableModel );
880
        String columnNames[] = WorkGroup.getFieldNames();
882     for ( int i = 0 ; i < columnNames.length; i++ )
        {
884        tableModel.addColumn( columnNames[i] );
        }
886  }
     /**
888   * Insert the method's description here.
     * Creation date: (16-08-00 18:15:00)
890   */
```

```
public void initWindow()
{
}
/**
 * Insert the method's description here.
 * Creation date: (06-07-00 12:48:00)
 */
public void joinGroup()
{
   int selectedIndex = getScrollPaneTable().getSelectedRow();
   // if no row is selected
   if ( selectedIndex == -1 )
   {
      MessageBox messageBox = new MessageBox();
      messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_JOIN_MESSAGE() );
      messageBox.show();
      return;
   }
   WorkGroup workGroup = (WorkGroup) groupDirectoryValues[ selectedIndex ];
   URLString url = workGroup.getSessionURL();
   Session session;
   Channel channel;
   String clientNames[] = null;
   try
   {
      session = SessionFactory.createSession( getClientApplication().getClientManager().
           getGroupManagementClient(), url, false );
      channel = session.createChannel( getClientApplication().getClientManager().
           getGroupManagementClient(),
                                workGroup.getWorkGroupChannelID(),
                                true, true, false);
      clientNames = channel.listClientNames();
   }
   // the below is a remnant from an attempt to resolve the problem using a different method than a for
        loop cycling
   // through the member names, comparing each to the unique name of this client.
   catch ( NameInUseException NIUE )
   {
      MessageBox messageBox = new MessageBox();
      messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
      messageBox.show();
      return;
   }
   catch ( JSDTException JSDTE )
   {
      getClientApplication().getClientManager().handleJSDTException( JSDTE );
   }
   for ( int i = 0 ; i < clientNames.length; i++ )
   {
      if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
           getNAME() )))
      {
         MessageBox messageBox = new MessageBox();
         messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
         messageBox.show();
         return;
      }
   }
/* for ( int i = 0; i < clientNames.length; i++ )
   {
      // System.out.println( "Clientname: " + clientNames[i] );
      // if client is already joined to the selected group
      if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
           getNAME() )));
      {
      }
```

```
952     }
    */
954     WorkGroup activeGroup = getClientApplication().getClientManager().getClientInfo().getActiveGroup();
        getClientApplication().getClientManager().disconnectFromGroup( activeGroup );
956     getClientApplication().getClientManager().connectToGroup( workGroup );
    }
958 /**
     * main entrypoint - starts the part when it is run as an application
960  * @param args java.lang.String[]
     */
962 public static void main(java.lang.String[] args) {
        try {
964         GroupDirectory aGroupDirectory;
            aGroupDirectory = new GroupDirectory();
966         aGroupDirectory.setModal(true);
            aGroupDirectory.addWindowListener(new java.awt.event.WindowAdapter() {
968             public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
970             };
            });
972         aGroupDirectory.setVisible(true);
        } catch (Throwable exception) {
974         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
            exception.printStackTrace(System.out);
976     }
    }
978 /**
     * Insert the method's description here.
980  * Creation date: (10-08-00 15:41:17)
     * @param newClientApplication clientapp.ClientApplication
982  */
    public void setClientApplication(ClientApplication newClientApplication) {
984     clientApplication = newClientApplication;
    }
986 /**
     * Insert the method's description here.
988  * Creation date: (26-07-00 21:09:59)
     * @param newGroupDirectoryValues peerviewmisc.WorkGroup[]
990  */
    public void setGroupDirectoryValues(peerviewmisc.WorkGroup[] newGroupDirectoryValues) {
992     groupDirectoryValues = newGroupDirectoryValues;
    }
994 /**
     * Insert the method's description here.
996  * Creation date: (04-07-00 19:38:32)
     * @param newTableModel javax.swing.table.DefaultTableModel
998  */
    public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
1000    tableModel = newTableModel;
    }
1002 /**
     * Insert the method's description here.
1004 * Creation date: (03-07-00 21:00:14)
     * @param newWorkGroup peerviewmisc.WorkGroup
1006 */
    public void setWorkGroup(peerviewmisc.WorkGroup newWorkGroup) {
1008    workGroup = newWorkGroup;
    }
1010 /**
     * Update the table model underlying the group directory.
1012 * Creation date: (22-07-00 23:35:40)
     */
1014 public void updateTable()
    {
1016    if ( getClientApplication().getClientManager().getClientInfo().getGroups() == null )
        {
```

```
      disableButtons();
      getClientApplication().getClientManager().requestGroupsFromServer();
      return;
   }
   enableButtons();
   // clear the table
   tableModel.setNumRows( 0 );
   // update with contents of group directory
   Collection values = getClientApplication().getClientManager().getClientInfo().getGroups().values();
   Iterator iterator = values.iterator();
   groupDirectoryValues = new WorkGroup[ values.size() ];
   int i = 0;
   while ( iterator.hasNext() )
   {
      workGroup = (WorkGroup) iterator.next();
      Vector v = workGroup.getGroupAsVector();
      tableModel.addRow( v );
      groupDirectoryValues[i++] = workGroup;
   }
   // Distribute surplus space evenly
   getScrollPaneTable().sizeColumnsToFit( -1 );
   getScrollPaneTable().revalidate();
   getScrollPaneTable().repaint();
   getJScrollPane1().revalidate();
   getJScrollPane1().repaint();

}
}
```

Listing C.14: MessagePropertiesBox.java

```
package clientapp;

 import javax.swing.table.DefaultTableModel;
 import java.util.Vector;
 import java.awt.event.ActionListener;

 /**
 * This class implements the message properties box that can be activated by selecting an item from the
     topic tree in the
 * client application message area and then pressing the message properties button.
 * Creation date: (19-07-00 17:55:27)
 * @author:
 */
public class MessagePropertiesBox extends javax.swing.JDialog {

   public class CloseButtonHandler implements ActionListener
   {
      public void actionPerformed( java.awt.event.ActionEvent ae )
      {
         dispose();
      }
   }
   private javax.swing.JPanel ivjJDialogContentPane = null;
   private javax.swing.JScrollPane ivjJScrollPane1 = null;
   private javax.swing.JTable ivjScrollPaneTable = null;
   private javax.swing.JButton ivjCloseButton = null;
   private javax.swing.table.DefaultTableModel tableModel = new DefaultTableModel();
   private javax.swing.JPanel ivjButtonPanel = null;
   private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
/**
 * MessagePropertiesBox constructor comment.
 */
public MessagePropertiesBox() {
   super();
   initialize();
}
```

```
36  /**
     * MessagePropertiesBox constructor comment.
38   * @param owner java.awt.Dialog
     */
40  public MessagePropertiesBox(java.awt.Dialog owner) {
       super(owner);
42  }
    /**
44   * MessagePropertiesBox constructor comment.
     * @param owner java.awt.Dialog
46   * @param title java.lang.String
     */
48  public MessagePropertiesBox(java.awt.Dialog owner, String title) {
       super(owner, title);
50  }
    /**
52   * MessagePropertiesBox constructor comment.
     * @param owner java.awt.Dialog
54   * @param title java.lang.String
     * @param modal boolean
56   */
    public MessagePropertiesBox(java.awt.Dialog owner, String title, boolean modal) {
58     super(owner, title, modal);
    }
60  /**
     * MessagePropertiesBox constructor comment.
62   * @param owner java.awt.Dialog
     * @param modal boolean
64   */
    public MessagePropertiesBox(java.awt.Dialog owner, boolean modal) {
66     super(owner, modal);
    }
68  /**
     * MessagePropertiesBox constructor comment.
70   * @param owner java.awt.Frame
     */
72  public MessagePropertiesBox(java.awt.Frame owner) {
       super(owner);
74  }
    /**
76   * MessagePropertiesBox constructor comment.
     * @param owner java.awt.Frame
78   * @param title java.lang.String
     */
80  public MessagePropertiesBox(java.awt.Frame owner, String title) {
       super(owner, title);
82  }
    /**
84   * MessagePropertiesBox constructor comment.
     * @param owner java.awt.Frame
86   * @param title java.lang.String
     * @param modal boolean
88   */
    public MessagePropertiesBox(java.awt.Frame owner, String title, boolean modal) {
90     super(owner, title, modal);
    }
92  /**
     * MessagePropertiesBox constructor comment.
94   * @param owner java.awt.Frame
     * @param modal boolean
96   */
    public MessagePropertiesBox(java.awt.Frame owner, boolean modal) {
98     super(owner, modal);
    }
100 /**
     * Return the JPanel1 property value.
```

```
102   * @return javax.swing.JPanel
      */
104  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getButtonPanel() {
106     if (ivjButtonPanel == null) {
           try {
108           ivjButtonPanel = new javax.swing.JPanel();
              ivjButtonPanel.setName("ButtonPanel");
110           ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
              getButtonPanel().add(getCloseButton(), getCloseButton().getName());
112           // user code begin {1}
              // user code end
114        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
116           // user code end
              handleException(ivjExc);
118        }
        }
120     return ivjButtonPanel;
     }
122  /**
      * Return the ButtonPanelFlowLayout property value.
124   * @return java.awt.FlowLayout
      */
126  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private java.awt.FlowLayout getButtonPanelFlowLayout() {
128     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
        try {
130        /* Create part */
           ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
132        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
        } catch (java.lang.Throwable ivjExc) {
134        handleException(ivjExc);
        };
136     return ivjButtonPanelFlowLayout;
     }
138  /**
      * Return the OKButton property value.
140   * @return javax.swing.JButton
      */
142  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getCloseButton() {
144     if (ivjCloseButton == null) {
           try {
146           ivjCloseButton = new javax.swing.JButton();
              ivjCloseButton.setName("CloseButton");
148           ivjCloseButton.setMnemonic('c');
              ivjCloseButton.setText("Close");
150           // user code begin {1}
              ivjCloseButton.addActionListener( new CloseButtonHandler() );
152           // user code end
           } catch (java.lang.Throwable ivjExc) {
154           // user code begin {2}
              // user code end
156           handleException(ivjExc);
           }
158     }
        return ivjCloseButton;
160  }
     /**
162   * Return the JDialogContentPane property value.
      * @return javax.swing.JPanel
164   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
166  private javax.swing.JPanel getJDialogContentPane() {
        if (ivjJDialogContentPane == null) {
```

```
168        try {
           ivjJDialogContentPane = new javax.swing.JPanel();
170        ivjJDialogContentPane.setName("JDialogContentPane");
           ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
172        getJDialogContentPane().add(getButtonPanel(), "South");
           getJDialogContentPane().add(getJScrollPane1(), "Center");
174        // user code begin {1}
           // user code end
176        } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
178        // user code end
           handleException(ivjExc);
180        }
        }
182     return ivjJDialogContentPane;
     }
184  /**
      * Return the JScrollPane1 property value.
186   * @return javax.swing.JScrollPane
      */
188  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JScrollPane getJScrollPane1() {
190     if (ivjJScrollPane1 == null) {
        try {
192        ivjJScrollPane1 = new javax.swing.JScrollPane();
           ivjJScrollPane1.setName("JScrollPane1");
194        ivjJScrollPane1.setAutoscrolls(true);
           ivjJScrollPane1.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
196        ivjJScrollPane1.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
               );
           getJScrollPane1().setViewportView(getScrollPaneTable());
198        // user code begin {1}
           // user code end
200        } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
202        // user code end
           handleException(ivjExc);
204        }
        }
206     return ivjJScrollPane1;
     }
208  /**
      * Return the ScrollPaneTable property value.
210   * @return javax.swing.JTable
      */
212  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JTable getScrollPaneTable() {
214     if (ivjScrollPaneTable == null) {
        try {
216        ivjScrollPaneTable = new javax.swing.JTable();
           ivjScrollPaneTable.setName("ScrollPaneTable");
218        getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
           getJScrollPane1().getViewport().setBackingStoreEnabled(true);
220        ivjScrollPaneTable.setBackground(java.awt.Color.white);
           ivjScrollPaneTable.setForeground(java.awt.Color.black);
222        ivjScrollPaneTable.setGridColor(java.awt.SystemColor.desktop);
           ivjScrollPaneTable.setFont(new java.awt.Font("dialog", 0, 12));
224        ivjScrollPaneTable.setIntercellSpacing(new java.awt.Dimension(5, 5));
           ivjScrollPaneTable.setBounds(0, 0, 200, 200);
226        // user code begin {1}
           ivjScrollPaneTable.setModel( tableModel );
228        ivjScrollPaneTable.setSelectionMode( javax.swing.ListSelectionModel.SINGLE_SELECTION );
           ivjScrollPaneTable.setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
230
           // user code end
232        } catch (java.lang.Throwable ivjExc) {
```

```
            // user code begin {2}
234         // user code end
            handleException(ivjExc);
236       }
      }
238   return ivjScrollPaneTable;
    }
240 /**
     * Insert the method's description here.
242  * Creation date: (19-07-00 19:46:21)
     * @return javax.swing.table.DefaultTableModel
244  */
    public javax.swing.table.DefaultTableModel getTableModel() {
246   return tableModel;
    }
248 /**
     * Called whenever the part throws an exception.
250  * @param exception java.lang.Throwable
     */
252 private void handleException(java.lang.Throwable exception) {

254   /* Uncomment the following lines to print uncaught exceptions to stdout */
      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
256   // exception.printStackTrace(System.out);
    }
258 /**
     * Initialize the class.
260  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
262 private void initialize() {
      try {
264     // user code begin {1}
        // user code end
266     setName("MessagePropertiesBox");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
268     setSize(415, 257);
        setModal(true);
270     setTitle("Message␣properties");
        setContentPane(getJDialogContentPane());
272   } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
274   }
      // user code begin {2}
276   // default positioning puts the dialog at center of screen. The method used is a slight hack, but
          works fine.
      setLocationRelativeTo( getOwner() );
278   getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
      // user code end
280 }
    /**
282  * Insert the method's description here.
     * Creation date: (19-07-00 19:45:02)
284  * @param message peerviewmisc.Message
     */
286 public void initTable(peerviewmisc.Message message)
    {
288   // The below ought to be replaceable with calls to addColumn, but an attempt to do so didn't work as
          intended - perhaps a JDK bug ?
      Object fieldNames[] = message.getFIELD_NAMES();
290   Vector values = message.getDataAsVector();
      tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_NAME_COLUMN_HEADING() );
292   tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADING() );
      for ( int i = 0; i < fieldNames.length; i++ )
294   {
        Vector v = new Vector();
296     v.addElement( fieldNames[i] );
```

```
      v.addElement( values.get(i) );
      tableModel.addRow( v );
   }
   getScrollPaneTable().setColumnSelectionAllowed( false );
   getScrollPaneTable().setRowSelectionAllowed( false );
   getScrollPaneTable().setEnabled( false );
   getScrollPaneTable().setCellEditor( null );
   getScrollPaneTable().removeEditor();
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      MessagePropertiesBox aMessagePropertiesBox;
      aMessagePropertiesBox = new MessagePropertiesBox();
      aMessagePropertiesBox.setModal(true);
      aMessagePropertiesBox.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aMessagePropertiesBox.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 19:46:21)
 * @param newTableModel javax.swing.table.DefaultTableModel
 */
public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
   tableModel = newTableModel;
}
}
```

Listing C.15: MessageWindow.java

```
package clientapp;

 import java.awt.event.*;
 import javax.swing.event.*;

 /**
 * This class implements the message window that can be activated by double clicking in the message bar
     at the bottom of the
 * client application window.
 * Creation date: (16-08-00 16:44:45)
 * @author:
 */
public class MessageWindow extends javax.swing.JDialog {

   public class CloseItemHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         closeMessageWindow();
      }
   }

   public class SaveItemHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
```

```
26          saveMessageWindowContents();
         }
28    }
      private javax.swing.JMenuItem ivjCloseItem = null;
30    private javax.swing.JMenu ivjFileMenu = null;
      private javax.swing.JPanel ivjJDialogContentPane = null;
32    private javax.swing.JScrollPane ivjJScrollPane1 = null;
      private javax.swing.JSeparator ivjJSeparator1 = null;
34    private javax.swing.JMenuBar ivjMessageWindowJMenuBar = null;
      private javax.swing.JMenuItem ivjSaveItem = null;
36    private javax.swing.JTextArea ivjTextArea = null;
      private ClientApplication clientApplication = null;
38  /**
     * MessageWindow constructor comment.
40   */
    public MessageWindow() {
42     super();
       initialize();
44  }
    /**
46   * MessageWindow constructor comment.
     * @param owner java.awt.Dialog
48   */
    public MessageWindow(java.awt.Dialog owner) {
50     super(owner);
    }
52  /**
     * MessageWindow constructor comment.
54   * @param owner java.awt.Dialog
     * @param title java.lang.String
56   */
    public MessageWindow(java.awt.Dialog owner, String title) {
58     super(owner, title);
    }
60  /**
     * MessageWindow constructor comment.
62   * @param owner java.awt.Dialog
     * @param title java.lang.String
64   * @param modal boolean
     */
66  public MessageWindow(java.awt.Dialog owner, String title, boolean modal) {
       super(owner, title, modal);
68  }
    /**
70   * MessageWindow constructor comment.
     * @param owner java.awt.Dialog
72   * @param modal boolean
     */
74  public MessageWindow(java.awt.Dialog owner, boolean modal) {
       super(owner, modal);
76  }
    /**
78   * MessageWindow constructor comment.
     * @param owner java.awt.Frame
80   */
    public MessageWindow(java.awt.Frame owner) {
82     super(owner);
    }
84  /**
     * MessageWindow constructor comment.
86   * @param owner java.awt.Frame
     * @param title java.lang.String
88   */
    public MessageWindow(java.awt.Frame owner, String title) {
90     super(owner, title);
    }
```

```
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public MessageWindow(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * MessageWindow constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public MessageWindow(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * Insert the method's description here.
 * Creation date: (16-08-00 17:13:28)
 * @param string java.lang.String
 */
public void addString(String string)
{
   getTextArea().append( string + "\n" );
}
/**
 * Insert the method's description here.
 * Creation date: (16-08-00 17:51:13)
 */
public void closeMessageWindow()
{
   setVisible( false );
}
/**
 * Insert the method's description here.
 * Creation date: (29-10-00 17:08:48)
 * @return clientapp.ClientApplication
 */
public ClientApplication getClientApplication() {
   return clientApplication;
}
/**
 * Return the CloseItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getCloseItem() {
   if (ivjCloseItem == null) {
      try {
         ivjCloseItem = new javax.swing.JMenuItem();
         ivjCloseItem.setName("CloseItem");
         ivjCloseItem.setMnemonic('C');
         ivjCloseItem.setText("Close");
         // user code begin {1}
         ivjCloseItem.addActionListener( new CloseItemHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjCloseItem;
}
/**
```

```
158   * Return the FileMenu property value.
      * @return javax.swing.JMenu
160   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
162  private javax.swing.JMenu getFileMenu() {
        if (ivjFileMenu == null) {
164        try {
              ivjFileMenu = new javax.swing.JMenu();
166           ivjFileMenu.setName("FileMenu");
              ivjFileMenu.setMnemonic('F');
168           ivjFileMenu.setText("File");
              ivjFileMenu.add(getSaveItem());
170           ivjFileMenu.add(getJSeparator1());
              ivjFileMenu.add(getCloseItem());
172           // user code begin {1}
              // user code end
174        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
176           // user code end
              handleException(ivjExc);
178        }
        }
180     return ivjFileMenu;
     }
182  /**
      * Return the JDialogContentPane property value.
184   * @return javax.swing.JPanel
      */
186  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJDialogContentPane() {
188     if (ivjJDialogContentPane == null) {
           try {
190           ivjJDialogContentPane = new javax.swing.JPanel();
              ivjJDialogContentPane.setName("JDialogContentPane");
192           ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
              getJDialogContentPane().add(getJScrollPane1(), "Center");
194           // user code begin {1}
              // user code end
196        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
198           // user code end
              handleException(ivjExc);
200        }
        }
202     return ivjJDialogContentPane;
     }
204  /**
      * Return the JScrollPane1 property value.
206   * @return javax.swing.JScrollPane
      */
208  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JScrollPane getJScrollPane1() {
210     if (ivjJScrollPane1 == null) {
           try {
212           ivjJScrollPane1 = new javax.swing.JScrollPane();
              ivjJScrollPane1.setName("JScrollPane1");
214           getJScrollPane1().setViewportView(getTextArea());
              // user code begin {1}
216           // user code end
           } catch (java.lang.Throwable ivjExc) {
218           // user code begin {2}
              // user code end
220           handleException(ivjExc);
           }
222     }
        return ivjJScrollPane1;
```

```
224  }
     /**
226   * Return the JSeparator1 property value.
      * @return javax.swing.JSeparator
228   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
230  private javax.swing.JSeparator getJSeparator1() {
        if (ivjJSeparator1 == null) {
232        try {
              ivjJSeparator1 = new javax.swing.JSeparator();
234           ivjJSeparator1.setName("JSeparator1");
              // user code begin {1}
236           // user code end
           } catch (java.lang.Throwable ivjExc) {
238           // user code begin {2}
              // user code end
240           handleException(ivjExc);
           }
242     }
        return ivjJSeparator1;
244  }
     /**
246   * Return the MessageWindowJMenuBar property value.
      * @return javax.swing.JMenuBar
248   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
250  private javax.swing.JMenuBar getMessageWindowJMenuBar() {
        if (ivjMessageWindowJMenuBar == null) {
252        try {
              ivjMessageWindowJMenuBar = new javax.swing.JMenuBar();
254           ivjMessageWindowJMenuBar.setName("MessageWindowJMenuBar");
              ivjMessageWindowJMenuBar.add(getFileMenu());
256           // user code begin {1}
              // user code end
258        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
260           // user code end
              handleException(ivjExc);
262        }
        }
264     return ivjMessageWindowJMenuBar;
     }
266  /**
      * Return the SaveItem property value.
268   * @return javax.swing.JMenuItem
      */
270  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JMenuItem getSaveItem() {
272     if (ivjSaveItem == null) {
           try {
274           ivjSaveItem = new javax.swing.JMenuItem();
              ivjSaveItem.setName("SaveItem");
276           ivjSaveItem.setMnemonic('S');
              ivjSaveItem.setText("Save");
278           // user code begin {1}
              ivjSaveItem.addActionListener( new SaveItemHandler() );
280           // user code end
           } catch (java.lang.Throwable ivjExc) {
282           // user code begin {2}
              // user code end
284           handleException(ivjExc);
           }
286     }
        return ivjSaveItem;
288  }
     /**
```

```
290   * Return the TextArea property value.
      * @return javax.swing.JTextPane
292   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
294  private javax.swing.JTextArea getTextArea() {
        if (ivjTextArea == null) {
296        try {
              ivjTextArea = new javax.swing.JTextArea();
298           ivjTextArea.setName("TextArea");
              ivjTextArea.setToolTipText("Messages␣generated␣by␣the␣peerview␣client␣application.");
300           ivjTextArea.setBounds(0, 0, 7, 6);
              // user code begin {1}
302           // user code end
           } catch (java.lang.Throwable ivjExc) {
304           // user code begin {2}
              // user code end
306           handleException(ivjExc);
           }
308     }
        return ivjTextArea;
310  }
     /**
312   * Called whenever the part throws an exception.
      * @param exception java.lang.Throwable
314   */
     private void handleException(java.lang.Throwable exception) {
316
        /* Uncomment the following lines to print uncaught exceptions to stdout */
318     // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
320  }
     /**
322   * Initialize the class.
      */
324  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
326     try {
           // user code begin {1}
328        // user code end
           setName("MessageWindow");
330        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setTitle("Message␣window");
332        setSize(512, 479);
           setVisible(false);
334        setJMenuBar(getMessageWindowJMenuBar());
           setContentPane(getJDialogContentPane());
336     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
338     }
        // user code begin {2}
340     getTextArea().setEditable( false );
        getTextArea().setLineWrap( false );
342     setDefaultCloseOperation( javax.swing.JDialog.HIDE_ON_CLOSE );
        setLocationRelativeTo( getOwner() );
344     // user code end
     }
346  /**
      * main entrypoint - starts the part when it is run as an application
348   * @param args java.lang.String[]
      */
350  public static void main(java.lang.String[] args) {
        try {
352        MessageWindow aMessageWindow;
           aMessageWindow = new MessageWindow();
354        aMessageWindow.setModal(true);
           aMessageWindow.addWindowListener(new java.awt.event.WindowAdapter() {
```

```
356         public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
358         };
         });
360      aMessageWindow.setVisible(true);
      } catch (Throwable exception) {
362      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
364   }
   }
366 /**
    * Save the contents of the message window to disk.
368 * Creation date: (29-10-00 16:34:58)
    */
370 public void saveMessageWindowContents()
   {
372   try
      {
374      java.io.FileOutputStream fos = new java.io.FileOutputStream( ClientInfo.getPreferences().
            getProperty( ClientConstants.getMESSAGE_LOG_FILENAME() ));
         java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream( fos );
376
         oos.writeUTF( getTextArea().getText() );
378      oos.flush();
         oos.close();
380
      }
382   catch ( Exception E )
      {
384      System.err.println( ClientConstants.getMESSAGE_LOG_WRITE_ERROR() );
      }
386
   }
388 /**
    * Insert the method's description here.
390 * Creation date: (29-10-00 17:08:48)
    * @param newClientApplication clientapp.ClientApplication
392 */
   public void setClientApplication(ClientApplication newClientApplication) {
394   clientApplication = newClientApplication;
   }
396 }
```

Listing C.16: PanoramaLayout.java

```
   package clientapp;
 2
   /**
 4  * Abstract superclass for client panorama layout managers.
    * Creation date: (22-06-00 11:30:49)
 6  * @author:
    */
 8  import java.awt.Dimension;

10 public abstract class PanoramaLayout
   {
12    protected java.awt.Dimension preferredDocumentSize = new Dimension();
      protected int xSpacing;
14    protected int ySpacing;
      protected java.awt.Dimension previousPlacement = null;
16    protected int horizontalIndex = 0;
      protected int verticalIndex = 0;
18 /**
    * PanoramaLayout constructor comment.
20  */
   public PanoramaLayout( )
22 {
```

```
      preferredDocumentSize = new Dimension(  Integer.parseInt( (String) ClientInfo.getPreferences().
          getProperty(
                                  ClientConstants.getDEFAULT_DOCUMENT_WIDTH() ) ),
                                Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
                                  ClientConstants.getDEFAULT_DOCUMENT_HEIGHT() )));
      xSpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
                          ClientConstants.getDEFAULT_HORIZONTAL_DOCUMENT_SPACING() ));
      ySpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
                          ClientConstants.getDEFAULT_VERTICAL_DOCUMENT_SPACING() ));
   }
   /**
    * Insert the method's description here.
    * Creation date: (22-06-00 11:50:28)
    * @return java.awt.Dimension
    */
   public abstract Dimension computePlacement() throws Exception;
   /**
    * Insert the method's description here.
    * Creation date: (01-08-00 15:42:44)
    * @return int
    */
   public int getHorizontalIndex() {
      return horizontalIndex;
   }
   /**
    * Insert the method's description here.
    * Creation date: (01-08-00 19:42:20)
    * @return java.awt.Rectangle
    */
   public abstract java.awt.Rectangle getOverviewRectangle();
   /**
    * Insert the method's description here.
    * Creation date: (22-06-00 11:32:36)
    * @return java.awt.Dimension
    */
   public java.awt.Dimension getPreferredDocumentSize() {
      return preferredDocumentSize;
   }
   /**
    * Insert the method's description here.
    * Creation date: (22-06-00 12:02:54)
    * @return java.awt.Dimension
    */
   public java.awt.Dimension getPreviousPlacement() {
      return previousPlacement;
   }
   /**
    * Insert the method's description here.
    * Creation date: (01-08-00 15:42:53)
    * @return int
    */
   public int getVerticalIndex() {
      return verticalIndex;
   }
   /**
    * Insert the method's description here.
    * Creation date: (22-06-00 11:37:05)
    * @return int
    */
   public int getXSpacing() {
      return xSpacing;
   }
   /**
    * Insert the method's description here.
    * Creation date: (22-06-00 11:37:25)
    * @return int
```

```
 */
public int getYSpacing() {
   return ySpacing;
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 22:22:43)
 */
public abstract void resetLayout();
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 15:42:44)
 * @param newHorizontalIndex int
 */
public void setHorizontalIndex(int newHorizontalIndex) {
   horizontalIndex = newHorizontalIndex;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:32:36)
 * @param newPreferredDocumentSize java.awt.Dimension
 */
public void setPreferredDocumentSize(java.awt.Dimension newPreferredDocumentSize) {
   preferredDocumentSize = newPreferredDocumentSize;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 12:02:54)
 * @param newPreviousPlacement java.awt.Dimension
 */
public void setPreviousPlacement(java.awt.Dimension newPreviousPlacement) {
   previousPlacement = newPreviousPlacement;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 15:42:53)
 * @param newVerticalIndex int
 */
public void setVerticalIndex(int newVerticalIndex) {
   verticalIndex = newVerticalIndex;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:37:05)
 * @param newXSpacing int
 */
public void setXSpacing(int newXSpacing) {
   xSpacing = newXSpacing;
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 11:37:25)
 * @param newYSpacing int
 */
public void setYSpacing(int newYSpacing) {
   ySpacing = newYSpacing;
}
}
```

Listing C.17: PeerViewClient.java

```
package clientapp;

import com.sun.media.jsdt.*;
import com.sun.media.jsdt.event.*;
/**
 * Insert the type's description here.
```

```
 * Creation date: (29-06-00 20:26:21)
 * @author:
 */
public class PeerViewClient implements Client, ChannelConsumer {
   private java.lang.String name;
/**
 * PeerViewClient constructor comment.
 */
public PeerViewClient(String nameArg)
{
   super();
   name = nameArg;
}
/**
 * authenticate method comment.
 */
public Object authenticate(AuthenticationInfo arg1) {
   return null;
}
/**
 * Insert the method's description here.
 * Creation date: (30-06-00 09:42:34)
 * @param date com.sun.media.jsdt.Data
 */
public synchronized void dataReceived(Data data)
{
   // System.out.println( data.getDataAsString() );
}
/**
 * Insert the method's description here.
 * Creation date: (30-06-00 09:00:06)
 * @return java.lang.String
 */
public java.lang.String getName()
{
   return name;
}
/**
 * Insert the method's description here.
 * Creation date: (30-06-00 09:00:06)
 * @param newName java.lang.String
 */
public void setName(java.lang.String newName) {
   name = newName;
}
}
```

Listing C.18: PreferencesDialog.java

```
package clientapp;

import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.util.Properties;
import java.io.*;
import java.awt.event.*;
import java.util.Hashtable;
import java.util.HashSet;
import java.util.*;
import peerviewmisc.*;

/** This class implements the preferences dialog and its associated panes.
*/
public class PreferencesDialog extends JDialog
{
   public class Changer implements ChangeListener
```

```
{
   public void stateChanged( ChangeEvent ce )
   {
      // System.out.println( getAnimationSpeedSlider().getValue() );
   }
};

public class OKHandler implements ActionListener
{
   public void actionPerformed (ActionEvent ae)
   {
      submitAndClose();
   }
}

public class CancelHandler implements ActionListener
{
   public void actionPerformed (ActionEvent ae)
   {
      cancelAndClose();
   }
}

public class ConnectHandler implements ActionListener
{
   public void actionPerformed (ActionEvent ae)
   {
      connectToServer();
   }
}

public class CustomizeHandler implements ActionListener
{
   public void actionPerformed( ActionEvent ae )
   {
      customizeLayout();
   }
};

private JPanel ivjJDialogContentPane = null;
private JTabbedPane ivjJTabbedPane1 = null;
private JPanel ivjDisplayPage = null;
private JLabel ivjNameLabel = null;
private JPanel ivjPersonalPage = null;
private JLabel ivjSignatureLabel = null;
private JButton ivjCancelButton = null;
private JPanel ivjJPanel1 = null;
private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
private JButton ivjOKButton = null;
private JScrollPane ivjJScrollPane1 = null;
private ClientApplication clientApplication = null;
private JButton ivjCustomizeButton = null;
private JComboBox ivjDisplayQualityBox = null;
private JLabel ivjDisplayQualityLabel = null;
private JComboBox ivjLayoutSchemeBox = null;
private JLabel ivjLayoutSchemeLabel = null;
IvjEventHandler ivjEventHandler = new IvjEventHandler();
private JTextField ivjNameTextField = null;
private JTextPane ivjSignaturePane = null;
private JLabel ivjServerName = null;
private JPanel ivjServerPage = null;
private JLabel ivjServerPortLabel = null;
private JTextField ivjServerNameField = null;
private JTextField ivjServerPortField = null;
private JLabel ivjConnectionType = null;
private JComboBox ivjConnectionTypeBox = null;
```

```
    private JTextField ivjUpdateIntervalField = null;
    private JLabel ivjUpdateIntervalLabel = null;
    private JButton ivjConnectButton = null;
    private JLabel ivjAnimationSpeedLabel = null;
    private JLabel ivjFastLabel = null;
    private JLabel ivjSlowLabel = null;
    private JSlider ivjAnimationSpeedSlider = null;

class IvjEventHandler implements java.awt.event.ActionListener {
      public void actionPerformed(java.awt.event.ActionEvent e) {
         if (e.getSource() == PreferencesDialog.this.getCancelButton())
            connEtoM1(e);
      };
   };
/**
 * PreferencesDialog constructor comment.
 */
public PreferencesDialog() {
   super();
   initialize();
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Dialog
 */
public PreferencesDialog(java.awt.Dialog owner) {
   super(owner);
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public PreferencesDialog(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public PreferencesDialog(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public PreferencesDialog(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Frame
 */
public PreferencesDialog(java.awt.Frame owner) {
   super(owner);
}
/**
 * PreferencesDialog constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public PreferencesDialog(java.awt.Frame owner, String title) {
```

```
      super(owner, title);
152 }
    /**
154  * PreferencesDialog constructor comment.
     * @param owner java.awt.Frame
156  * @param title java.lang.String
     * @param modal boolean
158  */
    public PreferencesDialog(java.awt.Frame owner, String title, boolean modal) {
160    super(owner, title, modal);
    }
162 /**
     * PreferencesDialog constructor comment.
164  * @param owner java.awt.Frame
     * @param modal boolean
166  */
    public PreferencesDialog(java.awt.Frame owner, boolean modal) {
168    super(owner, modal);
    }
170 /**
     * Insert the method's description here.
172  * Creation date: (05-08-00 20:27:21)
     */
174 public void cancelAndClose()
    {
176    dispose();
    }
178 /**
     * Insert the method's description here.
180  * Creation date: (10-08-00 20:27:16)
     */
182 public void connectToServer()
    {
184    ConfirmationBox confirmationBox = new ConfirmationBox();
       confirmationBox.setText( ClientConstants.getCONFIRM_CONNECT_TO_SERVER() );
186    confirmationBox.show();
       if ( confirmationBox.isConfirmed() )
188    {
          getClientApplication().getClientManager().connectToGroupManagement();
190    }
    }
192 /**
     * connEtoM1: (CancelButton.action.actionPerformed(java.awt.event.ActionEvent) --> PreferencesDialog.
         dispose()V)
194  * @param arg1 java.awt.event.ActionEvent
     */
196 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM1(java.awt.event.ActionEvent arg1) {
198    try {
          // user code begin {1}
200       // user code end
          this.dispose();
202       // user code begin {2}
          // user code end
204    } catch (java.lang.Throwable ivjExc) {
          // user code begin {3}
206       // user code end
          handleException(ivjExc);
208    }
    }
210 /**
     * Insert the method's description here.
212  * Creation date: (06-09-00 02:13:19)
     */
214 public void customizeLayout()
    {
```

```
216     getClientApplication().getPanoramaPanel().getLayoutManager().show( this );
        getClientApplication().getPanoramaPanel().getLayoutManager().updateSettings( ClientInfo.
            getPreferences() );
218  }
     /**
220   * Return the AnimationSpeedLabel property value.
      * @return javax.swing.JLabel
222   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
224  private javax.swing.JLabel getAnimationSpeedLabel() {
        if (ivjAnimationSpeedLabel == null) {
226        try {
              ivjAnimationSpeedLabel = new javax.swing.JLabel();
228           ivjAnimationSpeedLabel.setName("AnimationSpeedLabel");
              ivjAnimationSpeedLabel.setText("Animation␣speed:");
230           // user code begin {1}
              // user code end
232        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
234           // user code end
              handleException(ivjExc);
236        }
        }
238     return ivjAnimationSpeedLabel;
     }
240  /**
      * Return the JSlider1 property value.
242   * @return javax.swing.JSlider
      */
244  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JSlider getAnimationSpeedSlider() {
246     if (ivjAnimationSpeedSlider == null) {
        try {
248        ivjAnimationSpeedSlider = new javax.swing.JSlider();
           ivjAnimationSpeedSlider.setName("AnimationSpeedSlider");
250        // user code begin {1}
           ivjAnimationSpeedSlider.setMinimum( ClientConstants.getANIMATION_MIN_SPEED() );
252        ivjAnimationSpeedSlider.setMaximum( ClientConstants.getANIMATION_MAX_SPEED() );
           ivjAnimationSpeedSlider.setPaintTicks( true );
254        ivjAnimationSpeedSlider.setPaintTrack( true );
           ivjAnimationSpeedSlider.addChangeListener( new Changer() );
256        // user code end
           } catch (java.lang.Throwable ivjExc) {
258        // user code begin {2}
           // user code end
260        handleException(ivjExc);
        }
262     }
        return ivjAnimationSpeedSlider;
264  }
     /**
266   * Return the CancelButton property value.
      * @return javax.swing.JButton
268   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
270  private javax.swing.JButton getCancelButton() {
        if (ivjCancelButton == null) {
272        try {
              ivjCancelButton = new javax.swing.JButton();
274           ivjCancelButton.setName("CancelButton");
              ivjCancelButton.setText("Cancel");
276           // user code begin {1}
              ivjCancelButton.addActionListener( new CancelHandler() );
278           // user code end
        } catch (java.lang.Throwable ivjExc) {
280        // user code begin {2}
```

```
            // user code end
282         handleException(ivjExc);
         }
284   }
      return ivjCancelButton;
286 }
    /**
288  * Insert the method's description here.
     * Creation date: (25-06-00 20:14:31)
290  * @return clientapp.ClientApplication
     */
292 public ClientApplication getClientApplication() {
      return clientApplication;
294 }
    /**
296  * Return the ConnectButton property value.
     * @return javax.swing.JButton
298  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
300 private javax.swing.JButton getConnectButton() {
      if (ivjConnectButton == null) {
302      try {
            ivjConnectButton = new javax.swing.JButton();
304         ivjConnectButton.setName("ConnectButton");
            ivjConnectButton.setMnemonic('c');
306         ivjConnectButton.setText("Connect␣to␣server");
            // user code begin {1}
308         ivjConnectButton.setEnabled( false );
            ivjConnectButton.addActionListener( new ConnectHandler() );
310         // user code end
         } catch (java.lang.Throwable ivjExc) {
312         // user code begin {2}
            // user code end
314         handleException(ivjExc);
         }
316   }
      return ivjConnectButton;
318 }
    /**
320  * Return the ConnectionType property value.
     * @return javax.swing.JLabel
322  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
324 private javax.swing.JLabel getConnectionType() {
      if (ivjConnectionType == null) {
326      try {
            ivjConnectionType = new javax.swing.JLabel();
328         ivjConnectionType.setName("ConnectionType");
            ivjConnectionType.setText("Connection␣type:");
330         // user code begin {1}
            // user code end
332      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
334         // user code end
            handleException(ivjExc);
336      }
      }
338   return ivjConnectionType;
    }
340 /**
     * Return the ConnectionTypeBox property value.
342  * @return javax.swing.JComboBox
     */
344 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JComboBox getConnectionTypeBox() {
346   if (ivjConnectionTypeBox == null) {
```

```
      try {
         ivjConnectionTypeBox = new javax.swing.JComboBox();
         ivjConnectionTypeBox.setName("ConnectionTypeBox");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjConnectionTypeBox;
}
/**
 * Return the CustomizeButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCustomizeButton() {
   if (ivjCustomizeButton == null) {
      try {
         ivjCustomizeButton = new javax.swing.JButton();
         ivjCustomizeButton.setName("CustomizeButton");
         ivjCustomizeButton.setMnemonic('u');
         ivjCustomizeButton.setText("Customize");
         // user code begin {1}
         ivjCustomizeButton.addActionListener( new CustomizeHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjCustomizeButton;
}
/**
 * Return the DisplayPage property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getDisplayPage() {
   if (ivjDisplayPage == null) {
      try {
         ivjDisplayPage = new javax.swing.JPanel();
         ivjDisplayPage.setName("DisplayPage");
         ivjDisplayPage.setLayout(new java.awt.GridBagLayout());

         java.awt.GridBagConstraints constraintsDisplayQualityLabel = new java.awt.GridBagConstraints();
         constraintsDisplayQualityLabel.gridx = 0; constraintsDisplayQualityLabel.gridy = 0;
         constraintsDisplayQualityLabel.gridwidth = 2;
         constraintsDisplayQualityLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsDisplayQualityLabel.weighty = 1.0;
         constraintsDisplayQualityLabel.insets = new java.awt.Insets(6, 6, 6, 6);
         getDisplayPage().add(getDisplayQualityLabel(), constraintsDisplayQualityLabel);

         java.awt.GridBagConstraints constraintsLayoutSchemeLabel = new java.awt.GridBagConstraints();
         constraintsLayoutSchemeLabel.gridx = 0; constraintsLayoutSchemeLabel.gridy = 1;
         constraintsLayoutSchemeLabel.gridwidth = 2;
         constraintsLayoutSchemeLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsLayoutSchemeLabel.weighty = 1.0;
         constraintsLayoutSchemeLabel.insets = new java.awt.Insets(6, 6, 6, 6);
         getDisplayPage().add(getLayoutSchemeLabel(), constraintsLayoutSchemeLabel);

         java.awt.GridBagConstraints constraintsDisplayQualityBox = new java.awt.GridBagConstraints();
         constraintsDisplayQualityBox.gridx = 2; constraintsDisplayQualityBox.gridy = 0;
```

```
        constraintsDisplayQualityBox.gridwidth = 2;
414     constraintsDisplayQualityBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
        constraintsDisplayQualityBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
416     constraintsDisplayQualityBox.weightx = 0.3;
        constraintsDisplayQualityBox.weighty = 1.0;
418     constraintsDisplayQualityBox.insets = new java.awt.Insets(6, 6, 6, 6);
        getDisplayPage().add(getDisplayQualityBox(), constraintsDisplayQualityBox);
420
        java.awt.GridBagConstraints constraintsLayoutSchemeBox = new java.awt.GridBagConstraints();
422     constraintsLayoutSchemeBox.gridx = 2; constraintsLayoutSchemeBox.gridy = 1;
        constraintsLayoutSchemeBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
424     constraintsLayoutSchemeBox.anchor = java.awt.GridBagConstraints.NORTHEAST;
        constraintsLayoutSchemeBox.weightx = 0.3;
426     constraintsLayoutSchemeBox.weighty = 1.0;
        constraintsLayoutSchemeBox.insets = new java.awt.Insets(6, 6, 6, 6);
428     getDisplayPage().add(getLayoutSchemeBox(), constraintsLayoutSchemeBox);

430     java.awt.GridBagConstraints constraintsCustomizeButton = new java.awt.GridBagConstraints();
        constraintsCustomizeButton.gridx = 3; constraintsCustomizeButton.gridy = 1;
432     constraintsCustomizeButton.anchor = java.awt.GridBagConstraints.NORTHEAST;
        constraintsCustomizeButton.insets = new java.awt.Insets(4, 4, 4, 4);
434     getDisplayPage().add(getCustomizeButton(), constraintsCustomizeButton);

436     java.awt.GridBagConstraints constraintsUpdateIntervalLabel = new java.awt.GridBagConstraints();
        constraintsUpdateIntervalLabel.gridx = 1; constraintsUpdateIntervalLabel.gridy = 2;
438     constraintsUpdateIntervalLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
        constraintsUpdateIntervalLabel.weighty = 1.0;
440     constraintsUpdateIntervalLabel.insets = new java.awt.Insets(4, 4, 4, 4);
        getDisplayPage().add(getUpdateIntervalLabel(), constraintsUpdateIntervalLabel);
442
        java.awt.GridBagConstraints constraintsUpdateIntervalField = new java.awt.GridBagConstraints();
444     constraintsUpdateIntervalField.gridx = 2; constraintsUpdateIntervalField.gridy = 2;
        constraintsUpdateIntervalField.anchor = java.awt.GridBagConstraints.NORTHWEST;
446     constraintsUpdateIntervalField.weightx = 1.0;
        constraintsUpdateIntervalField.weighty = 1.0;
448     constraintsUpdateIntervalField.ipadx = 100;
        constraintsUpdateIntervalField.insets = new java.awt.Insets(4, 4, 4, 4);
450     getDisplayPage().add(getUpdateIntervalField(), constraintsUpdateIntervalField);

452     java.awt.GridBagConstraints constraintsAnimationSpeedLabel = new java.awt.GridBagConstraints();
        constraintsAnimationSpeedLabel.gridx = 1; constraintsAnimationSpeedLabel.gridy = 4;
454     constraintsAnimationSpeedLabel.anchor = java.awt.GridBagConstraints.WEST;
        constraintsAnimationSpeedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
456     getDisplayPage().add(getAnimationSpeedLabel(), constraintsAnimationSpeedLabel);

458     java.awt.GridBagConstraints constraintsAnimationSpeedSlider = new java.awt.GridBagConstraints();
        constraintsAnimationSpeedSlider.gridx = 2; constraintsAnimationSpeedSlider.gridy = 4;
460     constraintsAnimationSpeedSlider.gridwidth = 2;
        constraintsAnimationSpeedSlider.fill = java.awt.GridBagConstraints.HORIZONTAL;
462     constraintsAnimationSpeedSlider.anchor = java.awt.GridBagConstraints.EAST;
        constraintsAnimationSpeedSlider.weightx = 1.0;
464     constraintsAnimationSpeedSlider.insets = new java.awt.Insets(4, 4, 4, 4);
        getDisplayPage().add(getAnimationSpeedSlider(), constraintsAnimationSpeedSlider);
466
        java.awt.GridBagConstraints constraintsSlowLabel = new java.awt.GridBagConstraints();
468     constraintsSlowLabel.gridx = 2; constraintsSlowLabel.gridy = 3;
        constraintsSlowLabel.anchor = java.awt.GridBagConstraints.WEST;
470     constraintsSlowLabel.insets = new java.awt.Insets(4, 4, 4, 4);
        getDisplayPage().add(getSlowLabel(), constraintsSlowLabel);
472
        java.awt.GridBagConstraints constraintsFastLabel = new java.awt.GridBagConstraints();
474     constraintsFastLabel.gridx = 3; constraintsFastLabel.gridy = 3;
        constraintsFastLabel.anchor = java.awt.GridBagConstraints.EAST;
476     constraintsFastLabel.insets = new java.awt.Insets(4, 4, 4, 4);
        getDisplayPage().add(getFastLabel(), constraintsFastLabel);
478     // user code begin {1}
```

```
            // user code end
480       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
482         // user code end
            handleException(ivjExc);
484       }
        }
486     return ivjDisplayPage;
    }
488 /**
     * Return the DisplayQualityBox property value.
490  * @return javax.swing.JComboBox
     */
492 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JComboBox getDisplayQualityBox() {
494     if (ivjDisplayQualityBox == null) {
        try {
496         ivjDisplayQualityBox = new javax.swing.JComboBox();
            ivjDisplayQualityBox.setName("DisplayQualityBox");
498         // user code begin {1}
            // user code end
500       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
502         // user code end
            handleException(ivjExc);
504       }
        }
506     return ivjDisplayQualityBox;
    }
508 /**
     * Return the DisplayTypeLabel property value.
510  * @return javax.swing.JLabel
     */
512 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getDisplayQualityLabel() {
514     if (ivjDisplayQualityLabel == null) {
        try {
516         ivjDisplayQualityLabel = new javax.swing.JLabel();
            ivjDisplayQualityLabel.setName("DisplayQualityLabel");
518         ivjDisplayQualityLabel.setText("Display␣quality:");
            // user code begin {1}
520         // user code end
        } catch (java.lang.Throwable ivjExc) {
522         // user code begin {2}
            // user code end
524         handleException(ivjExc);
        }
526     }
        return ivjDisplayQualityLabel;
528 }
    /**
530  * Return the FastLabel property value.
     * @return javax.swing.JLabel
532  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
534 private javax.swing.JLabel getFastLabel() {
        if (ivjFastLabel == null) {
536     try {
            ivjFastLabel = new javax.swing.JLabel();
538         ivjFastLabel.setName("FastLabel");
            ivjFastLabel.setText("Fast");
540         // user code begin {1}
            // user code end
542     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
544         // user code end
```

```
            handleException(ivjExc);
546         }
      }
548   return ivjFastLabel;
   }
550 /**
    * Return the JDialogContentPane property value.
552 * @return javax.swing.JPanel
    */
554 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJDialogContentPane() {
556   if (ivjJDialogContentPane == null) {
         try {
558         ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
560         ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getJTabbedPane1(), "Center");
562         getJDialogContentPane().add(getJPanel1(), "South");
            // user code begin {1}
564         // user code end
         } catch (java.lang.Throwable ivjExc) {
566         // user code begin {2}
            // user code end
568         handleException(ivjExc);
         }
570   }
      return ivjJDialogContentPane;
572 }
   /**
574 * Return the JPanel1 property value.
    * @return javax.swing.JPanel
576 */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
578 private javax.swing.JPanel getJPanel1() {
      if (ivjJPanel1 == null) {
580      try {
            ivjJPanel1 = new javax.swing.JPanel();
582         ivjJPanel1.setName("JPanel1");
            ivjJPanel1.setLayout(getJPanel1FlowLayout());
584         getJPanel1().add(getOKButton(), getOKButton().getName());
            getJPanel1().add(getCancelButton(), getCancelButton().getName());
586         // user code begin {1}
            // user code end
588      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
590         // user code end
            handleException(ivjExc);
592      }
      }
594   return ivjJPanel1;
   }
596 /**
    * Return the JPanel1FlowLayout property value.
598 * @return java.awt.FlowLayout
    */
600 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private java.awt.FlowLayout getJPanel1FlowLayout() {
602   java.awt.FlowLayout ivjJPanel1FlowLayout = null;
      try {
604      /* Create part */
         ivjJPanel1FlowLayout = new java.awt.FlowLayout();
606      ivjJPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
      } catch (java.lang.Throwable ivjExc) {
608      handleException(ivjExc);
      };
610   return ivjJPanel1FlowLayout;
```

```
    }
612 /**
     * Return the JScrollPane1 property value.
614  * @return javax.swing.JScrollPane
     */
616 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JScrollPane getJScrollPane1() {
618    if (ivjJScrollPane1 == null) {
          try {
620          ivjJScrollPane1 = new javax.swing.JScrollPane();
             ivjJScrollPane1.setName("JScrollPane1");
622          getJScrollPane1().setViewportView(getSignaturePane());
             // user code begin {1}
624          // user code end
          } catch (java.lang.Throwable ivjExc) {
626          // user code begin {2}
             // user code end
628          handleException(ivjExc);
          }
630    }
       return ivjJScrollPane1;
632 }
    /**
634  * Return the JTabbedPane1 property value.
     * @return javax.swing.JTabbedPane
636  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
638 private javax.swing.JTabbedPane getJTabbedPane1() {
       if (ivjJTabbedPane1 == null) {
640       try {
             ivjJTabbedPane1 = new javax.swing.JTabbedPane();
642          ivjJTabbedPane1.setName("JTabbedPane1");
             ivjJTabbedPane1.insertTab("Display", null, getDisplayPage(), null, 0);
644          ivjJTabbedPane1.insertTab("Personal", null, getPersonalPage(), null, 1);
             ivjJTabbedPane1.insertTab("Server", null, getServerPage(), null, 2);
646          // user code begin {1}
             // user code end
648       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
650          // user code end
             handleException(ivjExc);
652       }
       }
654    return ivjJTabbedPane1;
    }
656 /**
     * Return the JComboBox5 property value.
658  * @return javax.swing.JComboBox
     */
660 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JComboBox getLayoutSchemeBox() {
662    if (ivjLayoutSchemeBox == null) {
          try {
664          ivjLayoutSchemeBox = new javax.swing.JComboBox();
             ivjLayoutSchemeBox.setName("LayoutSchemeBox");
666          // user code begin {1}
             for (int i = 0; i < ClientConstants.getLAYOUT_SCHEMES().length; i++)
668          {
                ivjLayoutSchemeBox.addItem( ClientConstants.getLAYOUT_SCHEMES()[i] );
670          }
             // user code end
672       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
674          // user code end
             handleException(ivjExc);
676       }
```

```
      }
      return ivjLayoutSchemeBox;
   }
   /**
    * Return the TextColourLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getLayoutSchemeLabel() {
      if (ivjLayoutSchemeLabel == null) {
         try {
            ivjLayoutSchemeLabel = new javax.swing.JLabel();
            ivjLayoutSchemeLabel.setName("LayoutSchemeLabel");
            ivjLayoutSchemeLabel.setText("Layout␣scheme:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjLayoutSchemeLabel;
   }
   /**
    * Return the NameLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getNameLabel() {
      if (ivjNameLabel == null) {
         try {
            ivjNameLabel = new javax.swing.JLabel();
            ivjNameLabel.setName("NameLabel");
            ivjNameLabel.setText("Name:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjNameLabel;
   }
   /**
    * Return the JTextField1 property value.
    * @return javax.swing.JTextField
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextField getNameTextField() {
      if (ivjNameTextField == null) {
         try {
            ivjNameTextField = new javax.swing.JTextField();
            ivjNameTextField.setName("NameTextField");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjNameTextField;
   }
   /**
```

```
     * Return the OKButton property value.
744  * @return javax.swing.JButton
     */
746 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getOKButton() {
748    if (ivjOKButton == null) {
          try {
750          ivjOKButton = new javax.swing.JButton();
             ivjOKButton.setName("OKButton");
752          ivjOKButton.setText("OK");
             // user code begin {1}
754          // user code end
          } catch (java.lang.Throwable ivjExc) {
756          // user code begin {2}
             // user code end
758          handleException(ivjExc);
          }
760    }
       return ivjOKButton;
762 }
    /**
764  * Return the PersonalPage property value.
     * @return javax.swing.JPanel
766  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
768 private javax.swing.JPanel getPersonalPage() {
       if (ivjPersonalPage == null) {
770       try {
             ivjPersonalPage = new javax.swing.JPanel();
772          ivjPersonalPage.setName("PersonalPage");
             ivjPersonalPage.setLayout(new java.awt.GridBagLayout());
774
             java.awt.GridBagConstraints constraintsNameLabel = new java.awt.GridBagConstraints();
776          constraintsNameLabel.gridx = 0; constraintsNameLabel.gridy = 0;
             constraintsNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
778          constraintsNameLabel.weighty = 0.1;
             constraintsNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
780          getPersonalPage().add(getNameLabel(), constraintsNameLabel);

782          java.awt.GridBagConstraints constraintsSignatureLabel = new java.awt.GridBagConstraints();
             constraintsSignatureLabel.gridx = 0; constraintsSignatureLabel.gridy = 1;
784          constraintsSignatureLabel.gridwidth = 2;
             constraintsSignatureLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
786          constraintsSignatureLabel.weighty = 0.5;
             constraintsSignatureLabel.insets = new java.awt.Insets(6, 6, 6, 6);
788          getPersonalPage().add(getSignatureLabel(), constraintsSignatureLabel);

790          java.awt.GridBagConstraints constraintsNameTextField = new java.awt.GridBagConstraints();
             constraintsNameTextField.gridx = 1; constraintsNameTextField.gridy = 0;
792          constraintsNameTextField.gridwidth = 2;
             constraintsNameTextField.fill = java.awt.GridBagConstraints.HORIZONTAL;
794          constraintsNameTextField.anchor = java.awt.GridBagConstraints.NORTHEAST;
             constraintsNameTextField.weightx = 1.0;
796          constraintsNameTextField.weighty = 0.1;
             constraintsNameTextField.insets = new java.awt.Insets(6, 6, 6, 6);
798          getPersonalPage().add(getNameTextField(), constraintsNameTextField);

800          java.awt.GridBagConstraints constraintsJScrollPane1 = new java.awt.GridBagConstraints();
             constraintsJScrollPane1.gridx = 2; constraintsJScrollPane1.gridy = 1;
802          constraintsJScrollPane1.gridwidth = 2;
             constraintsJScrollPane1.fill = java.awt.GridBagConstraints.BOTH;
804          constraintsJScrollPane1.anchor = java.awt.GridBagConstraints.NORTHEAST;
             constraintsJScrollPane1.weightx = 0.5;
806          constraintsJScrollPane1.weighty = 0.5;
             constraintsJScrollPane1.insets = new java.awt.Insets(6, 6, 6, 6);
808          getPersonalPage().add(getJScrollPane1(), constraintsJScrollPane1);
```

```
            // user code begin {1}
810         // user code end
         } catch (java.lang.Throwable ivjExc) {
812         // user code begin {2}
            // user code end
814         handleException(ivjExc);
         }
816   }
      return ivjPersonalPage;
818 }
    /**
820  * Return the ServerName property value.
     * @return javax.swing.JLabel
822  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
824 private javax.swing.JLabel getServerName() {
      if (ivjServerName == null) {
826      try {
            ivjServerName = new javax.swing.JLabel();
828         ivjServerName.setName("ServerName");
            ivjServerName.setText("Server␣name:");
830         // user code begin {1}
            // user code end
832      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
834         // user code end
            handleException(ivjExc);
836      }
      }
838   return ivjServerName;
    }
840 /**
     * Return the JTextField1 property value.
842  * @return javax.swing.JTextField
     */
844 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JTextField getServerNameField() {
846   if (ivjServerNameField == null) {
         try {
848         ivjServerNameField = new javax.swing.JTextField();
            ivjServerNameField.setName("ServerNameField");
850         // user code begin {1}
            // user code end
852      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
854         // user code end
            handleException(ivjExc);
856      }
      }
858   return ivjServerNameField;
    }
860 /**
     * Return the ServerPage property value.
862  * @return javax.swing.JPanel
     */
864 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getServerPage() {
866   if (ivjServerPage == null) {
         try {
868         ivjServerPage = new javax.swing.JPanel();
            ivjServerPage.setName("ServerPage");
870         ivjServerPage.setOpaque(true);
            ivjServerPage.setLayout(new java.awt.GridBagLayout());
872
            java.awt.GridBagConstraints constraintsServerName = new java.awt.GridBagConstraints();
874         constraintsServerName.gridx = 0; constraintsServerName.gridy = 0;
```

```
            constraintsServerName.anchor = java.awt.GridBagConstraints.NORTHWEST;
876         constraintsServerName.weighty = 1.0;
            constraintsServerName.insets = new java.awt.Insets(4, 4, 4, 4);
878         getServerPage().add(getServerName(), constraintsServerName);

880         java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
            constraintsServerNameField.gridx = 1; constraintsServerNameField.gridy = 0;
882         constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
            constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHWEST;
884         constraintsServerNameField.weightx = 1.0;
            constraintsServerNameField.weighty = 1.0;
886         constraintsServerNameField.insets = new java.awt.Insets(4, 4, 4, 4);
            getServerPage().add(getServerNameField(), constraintsServerNameField);
888
            java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
890         constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
            constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
892         constraintsServerPortLabel.weighty = 1.0;
            constraintsServerPortLabel.insets = new java.awt.Insets(4, 4, 4, 4);
894         getServerPage().add(getServerPortLabel(), constraintsServerPortLabel);

896         java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
            constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
898         constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsServerPortField.weightx = 1.0;
900         constraintsServerPortField.weighty = 1.0;
            constraintsServerPortField.ipadx = 100;
902         constraintsServerPortField.insets = new java.awt.Insets(4, 4, 4, 4);
            getServerPage().add(getServerPortField(), constraintsServerPortField);
904
            java.awt.GridBagConstraints constraintsConnectionTypeBox = new java.awt.GridBagConstraints();
906         constraintsConnectionTypeBox.gridx = 1; constraintsConnectionTypeBox.gridy = 2;
            constraintsConnectionTypeBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
908         constraintsConnectionTypeBox.weightx = 1.0;
            constraintsConnectionTypeBox.insets = new java.awt.Insets(4, 4, 4, 4);
910         getServerPage().add(getConnectionTypeBox(), constraintsConnectionTypeBox);

912         java.awt.GridBagConstraints constraintsConnectionType = new java.awt.GridBagConstraints();
            constraintsConnectionType.gridx = 0; constraintsConnectionType.gridy = 2;
914         constraintsConnectionType.insets = new java.awt.Insets(4, 4, 4, 4);
            getServerPage().add(getConnectionType(), constraintsConnectionType);
916
            java.awt.GridBagConstraints constraintsConnectButton = new java.awt.GridBagConstraints();
918         constraintsConnectButton.gridx = 1; constraintsConnectButton.gridy = 2;
            constraintsConnectButton.anchor = java.awt.GridBagConstraints.EAST;
920         constraintsConnectButton.insets = new java.awt.Insets(4, 4, 4, 4);
            getServerPage().add(getConnectButton(), constraintsConnectButton);
922         // user code begin {1}
            // user code end
924      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
926         // user code end
            handleException(ivjExc);
928      }
         }
930      return ivjServerPage;
      }
932   /**
       * Return the JTextField2 property value.
934    * @return javax.swing.JTextField
       */
936   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTextField getServerPortField() {
938      if (ivjServerPortField == null) {
            try {
940            ivjServerPortField = new javax.swing.JTextField();
```

```
            ivjServerPortField.setName("ServerPortField");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjServerPortField;
   }
   /**
    * Return the ServerPortLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getServerPortLabel() {
      if (ivjServerPortLabel == null) {
         try {
            ivjServerPortLabel = new javax.swing.JLabel();
            ivjServerPortLabel.setName("ServerPortLabel");
            ivjServerPortLabel.setText("Server␣port:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjServerPortLabel;
   }
   /**
    * Return the SignatureLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getSignatureLabel() {
      if (ivjSignatureLabel == null) {
         try {
            ivjSignatureLabel = new javax.swing.JLabel();
            ivjSignatureLabel.setName("SignatureLabel");
            ivjSignatureLabel.setText("Signature:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjSignatureLabel;
   }
   /**
    * Return the JTextPane1 property value.
    * @return javax.swing.JTextPane
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextPane getSignaturePane() {
      if (ivjSignaturePane == null) {
         try {
            ivjSignaturePane = new javax.swing.JTextPane();
            ivjSignaturePane.setName("SignaturePane");
            ivjSignaturePane.setBounds(0, 0, 308, 93);
            // user code begin {1}
            // user code end
```

```
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjSignaturePane;
   }
   /**
    * Return the SlowLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getSlowLabel() {
      if (ivjSlowLabel == null) {
         try {
            ivjSlowLabel = new javax.swing.JLabel();
            ivjSlowLabel.setName("SlowLabel");
            ivjSlowLabel.setText("Slow");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjSlowLabel;
   }
   /**
    * Return the UpdateIntervalField property value.
    * @return javax.swing.JTextField
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextField getUpdateIntervalField() {
      if (ivjUpdateIntervalField == null) {
         try {
            ivjUpdateIntervalField = new javax.swing.JTextField();
            ivjUpdateIntervalField.setName("UpdateIntervalField");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjUpdateIntervalField;
   }
   /**
    * Return the UpdateIntervalLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getUpdateIntervalLabel() {
      if (ivjUpdateIntervalLabel == null) {
         try {
            ivjUpdateIntervalLabel = new javax.swing.JLabel();
            ivjUpdateIntervalLabel.setName("UpdateIntervalLabel");
            ivjUpdateIntervalLabel.setText("Update␣interval␣(seconds):");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
```

```
        }
1074    }
        return ivjUpdateIntervalLabel;
1076 }
     /**
1078  * Called whenever the part throws an exception.
      * @param exception java.lang.Throwable
1080  */
     private void handleException(java.lang.Throwable exception) {
1082
        /* Uncomment the following lines to print uncaught exceptions to stdout */
1084    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
1086 }
     /**
1088  * Insert the method's description here.
      * Creation date: (05-08-00 17:19:26)
1090  */
     public void init()
1092 {
        initialize();
1094 }
     /**
1096  * Initializes connections
      * @exception java.lang.Exception The exception description.
1098  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
1100 private void initConnections() throws java.lang.Exception {
        // user code begin {1}
1102    // user code end
        getCancelButton().addActionListener(ivjEventHandler);
1104 }
     /**
1106  * Initialize the class.
      */
1108 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
1110    try {
           // user code begin {1}
1112       // user code end
           setName("PreferencesDialog");
1114       setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setSize(509, 298);
1116       setModal(true);
           setTitle("Preferences");
1118       setContentPane(getJDialogContentPane());
           initConnections();
1120    } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
1122    }
        // user code begin {2}
1124    getOKButton().addActionListener( new OKHandler() );
        // user code end
1126 }
     /**
1128  * main entrypoint - starts the part when it is run as an application
      * @param args java.lang.String[]
1130  */
     public static void main(java.lang.String[] args) {
1132    try {
           PreferencesDialog aPreferencesDialog;
1134       aPreferencesDialog = new PreferencesDialog( new ClientApplication() );
           aPreferencesDialog.setModal(true);
1136       aPreferencesDialog.addWindowListener(new java.awt.event.WindowAdapter() {
              public void windowClosing(java.awt.event.WindowEvent e) {
1138             System.exit(0);
```

```
            };
1140        });
            aPreferencesDialog.setVisible(true);
1142     } catch (Throwable exception) {
            System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
1144        exception.printStackTrace(System.out);
         }
1146  }
      /**
1148   * Insert the method's description here.
       * Creation date: (25-06-00 20:14:31)
1150   * @param newClientApplication clientapp.ClientApplication
       */
1152  public void setClientApplication(ClientApplication newClientApplication) {
         clientApplication = newClientApplication;
1154  }
      /**
1156   * Overridden show method of the super-class to ensure proper initialization before the box is displayed
           .
       * Creation date: (05-08-00 20:42:55)
1158   */
      public void show()
1160  {
         if ( getConnectionTypeBox().getItemCount() == 0)
1162     {
            for ( int i = 0; i < ClientConstants.getCONNECTION_TYPES().length; i++ )
1164        {
               getConnectionTypeBox().addItem( ClientConstants.getCONNECTION_TYPES()[i] );
1166        }
         }
1168     if ( getDisplayQualityBox().getItemCount() == 0)
         {
1170        Iterator iterator = ClientConstants.getDISPLAY_QUALITIES().keySet().iterator();
            while ( iterator.hasNext() )
1172        {
               getDisplayQualityBox().addItem( iterator.next() );
1174        }
         }
1176     Properties p = ClientInfo.getPreferences();
         getServerNameField().setText( (String)p.getProperty( ClientConstants.getSERVER_NAME() ));
1178     getServerPortField().setText( (String)p.getProperty( ClientConstants.getSERVER_PORT() ));
         getNameTextField().setText( (String)p.getProperty( ClientConstants.getAUTHOR_NAME() ));
1180     getSignaturePane().setText( (String)p.getProperty( ClientConstants.getSIGNATURE() ));
         getLayoutSchemeBox().setSelectedItem( p.getProperty( ClientConstants.getLAYOUT_SCHEME() ));
1182     getDisplayQualityBox().setSelectedItem( p.getProperty( ClientConstants.getDISPLAY_QUALITY() ));
         getConnectionTypeBox().setSelectedItem( p.getProperty( ClientConstants.getCONNECTION_TYPE() ));
1184     getUpdateIntervalField().setText( (String)p.getProperty( ClientConstants.getUPDATE_INTERVAL() ));
         getAnimationSpeedSlider().setValue( ClientConstants.getANIMATION_MAX_SPEED() -
1186                            Integer.parseInt( (String) p.getProperty( ClientConstants.
                                    getANIMATION_DURATION_IN_MILLISECS() )));
         // System.out.println( getAnimationSpeedSlider().getValue() );
1188     setLocationRelativeTo( getOwner() );
         super.show();
1190  }
      /**
1192   * Commit preferences from preferences dialog panes and dispose of the dialog box.
       * Creation date: (26-06-00 07:50:15)
1194   */
      public void submitAndClose()
1196  {
         Properties p = ClientInfo.getPreferences();
1198     // fetch values from dialog box and insert into Properties object
         if ( getDisplayQualityBox().getSelectedItem() != null )
1200     {
            p.setProperty( ClientConstants.getDISPLAY_QUALITY(), (String)getDisplayQualityBox().
                 getSelectedItem() );
```

```
    }
    if ( getLayoutSchemeBox().getSelectedItem() != null )
    {
       p.setProperty( ClientConstants.getLAYOUT_SCHEME(), (String)getLayoutSchemeBox().getSelectedItem()
            );
    }
    p.setProperty( ClientConstants.getAUTHOR_NAME(), (String)getNameTextField().getText() );
    p.setProperty( ClientConstants.getSIGNATURE(), (String)getSignaturePane().getText() );
    p.setProperty( ClientConstants.getSERVER_NAME(), (String)getServerNameField().getText() );
    p.setProperty( ClientConstants.getSERVER_PORT(), (String)getServerPortField().getText() );
    if ( getConnectionTypeBox().getSelectedItem() != null )
    {
       p.setProperty( ClientConstants.getCONNECTION_TYPE(), (String)getConnectionTypeBox().
            getSelectedItem() );
    }
    p.setProperty( ClientConstants.getUPDATE_INTERVAL(), (String)getUpdateIntervalField().getText() );
    // store the inverse of the slider value to have it interpreted as a temporal value
    p.setProperty(  ClientConstants.getANIMATION_DURATION_IN_MILLISECS(),
               String.valueOf( ClientConstants.getANIMATION_MAX_SPEED() - getAnimationSpeedSlider().
                    getValue() ));

    ClientInfo.setPreferences( p );
    ClientInfo.writePreferences();
    dispose();
  }
}
```

Listing C.19: PZCamera.java

```
package clientapp;

/**
 * This is a customized version of the JAZZ ZCamera class. I'm not satisfied with this class as it
     stands since the code
 * in render() is bit of a kludge but it'll have to do for now.
 * Creation date: (01-08-00 16:49:47)
 * @author:
 */
public class PZCamera extends edu.umd.cs.jazz.ZCamera {
   private boolean centering = false;
/**
 * PZCamera constructor comment.
 */
public PZCamera() {
   super();
}
/**
 * PZCamera constructor comment.
 * @param layer edu.umd.cs.jazz.ZLayerGroup
 * @param aSurface edu.umd.cs.jazz.ZDrawingSurface
 */
public PZCamera(edu.umd.cs.jazz.ZLayerGroup layer, edu.umd.cs.jazz.ZDrawingSurface aSurface) {
   super(layer, aSurface);
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 16:54:15)
 * @return boolean
 */
public boolean isCentering() {
   return centering;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 16:56:11)
 * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
 */
```

```
public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
{
   if ( isCentering() == true )
   {
      return;
   }
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 16:54:15)
 * @param newCentering boolean
 */
public void setCentering(boolean newCentering) {
   centering = newCentering;
}
}
```

Listing C.20: PZCameraListener.java

```
package clientapp;

/**
 * Simple extension of the JAZZ camera adapter.
 * Creation date: (13-07-00 12:16:07)
 * @author:
 */
public class PZCameraListener extends edu.umd.cs.jazz.event.ZCameraAdapter
{
/**
 * Insert the method's description here.
 * Creation date: (13-07-00 12:17:43)
 * @param cameraEvent edu.umd.cs.jazz.event.ZCameraEvent
 */
public void viewChanged(edu.umd.cs.jazz.event.ZCameraEvent cameraEvent)
{

}
}
```

Listing C.21: PZoomHandler.java

```
package clientapp;

/**
 * An extension of the JAZZ zoom handler. This solution is a bit kludgy and should be refined when
     convenient.
 * Creation date: (16-06-00 23:46:26)
 * @author:
 */
public class PZoomHandler extends edu.umd.cs.jazz.event.ZoomEventHandler {
   // kan ævre lidt suspekt at bruge en static variabel til alle swing komponenter (se pzswing.render)
         - hvad
   // sker hvis der er flere instanser af klienten
   private static boolean Zooming = false;
   private static java.awt.Rectangle targetRectangle = null;
/**
 * PZoomHandler constructor comment.
 * @param node edu.umd.cs.jazz.ZNode
 */
public PZoomHandler(edu.umd.cs.jazz.ZNode node) {
   super(node);
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 23:28:20)
 * @return java.awt.Rectangle
```

```
 */
public static java.awt.Rectangle getTargetRectangle() {
   return targetRectangle;
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 10:45:17)
 * @return boolean
 */
public static boolean isZooming() {
   return Zooming;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 23:28:20)
 * @param newTargetRectangle java.awt.Rectangle
 */
public static void setTargetRectangle(java.awt.Rectangle newTargetRectangle) {
   targetRectangle = newTargetRectangle;
}
/**
 * Insert the method's description here.
 * Creation date: (01-08-00 17:07:39)
 * @param value boolean
 */
public static void setZooming(boolean value)
{
   Zooming = value;
}
/**
 * Insert the method's description here.
 * Creation date: (16-06-00 23:48:13)
 */
public void startZooming()
{
   // System.out.println("startZooming");
   Zooming = true;
   super.startZooming();
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 00:59:48)
 */
public void stopZooming()
{
   // System.out.println("stopZooming");
   Zooming = false;
   super.stopZooming();
}
}
```

Listing C.22: PZPanEventHandler.java

```
package clientapp;

/**
 * Simple extension of the JAZZ pan event handler.
 * Creation date: (17-06-00 13:44:31)
 * @author:
 */
public class PZPanEventHandler extends edu.umd.cs.jazz.event.ZPanEventHandler {
   private static boolean Panning = false;
/**
 * PZPanEventHandler constructor comment.
 * @param node edu.umd.cs.jazz.ZNode
 */
public PZPanEventHandler(edu.umd.cs.jazz.ZNode node) {
```

```
      super(node);
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 13:45:49)
 * @return boolean
 */
public static boolean isPanning() {
   return Panning;
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 13:47:43)
 * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
 */
public void mousePressed(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent)
{
   Panning = true;
   super.mousePressed(mouseEvent);
}
/**
 * Insert the method's description here.
 * Creation date: (17-06-00 13:48:07)
 * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
 */
public void mouseReleased(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent)
{
   Panning = false;
   super.mouseReleased(mouseEvent);
}
}
```

Listing C.23: PZSwing.java

```
package clientapp;

/**
 * Customization of the JAZZ Swing component wrapper.
 * Creation date: (17-06-00 10:17:04)
 * @author:
 */
public class PZSwing extends edu.umd.cs.jazz.component.ZSwing
{
   private edu.umd.cs.jazz.ZVisualLeaf zoomingNode = null;
   private float scaleFactor = (float)1.0;
   private java.awt.geom.AffineTransform originalTransform = null;
   private static boolean updating = false;
/**
 * PZSwing constructor comment.
 * @param zbc edu.umd.cs.jazz.util.ZCanvas
 * @param component javax.swing.JComponent
 */
public PZSwing(edu.umd.cs.jazz.util.ZCanvas zbc, javax.swing.JComponent component) {
   super(zbc, component);
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 17:50:17)
 * @return java.awt.geom.AffineTransform
 */
public java.awt.geom.AffineTransform getOriginalTransform() {
   return originalTransform;
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 17:43:52)
 * @return float
```

```
34   */
     public float getScaleFactor() {
36     return scaleFactor;
     }
38   /**
      * Insert the method's description here.
40    * Creation date: (30-07-00 17:26:06)
      * @return edu.umd.cs.jazz.ZVisualLeaf
42    */
     public edu.umd.cs.jazz.ZVisualLeaf getZoomingNode() {
44     return zoomingNode;
     }
46   /**
      * Insert the method's description here.
48    * Creation date: (07-09-00 23:31:48)
      * @return boolean
50    */
     public static boolean isUpdating() {
52     return updating;
     }
54   /**
      * Insert the method's description here.
56    * Creation date: (17-06-00 10:24:09)
      * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
58    */
     public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
60   {
       // don't render if updating - for efficiency and safety.
62     if ( isUpdating() )
       {
64       return;
       }
66     // skal smukkeseres
       if ( PZoomHandler.isZooming() || PZPanEventHandler.isPanning() )
68     {  paintAsGreek( renderContext.getGraphics2D() );
         return;
70     }
       // if a centering animation is in progress, paint as grey "blobs" rather than fully rendered graphics
72     if ( PZoomHandler.getTargetRectangle() != null)
       {  // if the animation is ongoing
74       //System.out.println( renderContext.getRenderingCamera().getViewBounds() );
         //System.out.println( PZoomHandler.getTargetRectangle() );
76       int currentWidth = (int)renderContext.getRenderingCamera().getViewBounds().getWidth();
         int targetWidth = (int)PZoomHandler.getTargetRectangle().getWidth();
78       int currentHeight = (int)renderContext.getRenderingCamera().getViewBounds().getHeight();
         int targetHeight = (int)PZoomHandler.getTargetRectangle().getHeight();
80       // AT ØGRE: Udskift ånedenstende magiske tal med konstant - eksperimenter for at finde en passende
               ævrdi
         if ( ( (currentWidth < ( targetWidth - 3 ))|| (currentWidth > ( targetWidth + 3 )))&&
82           ( (currentHeight < ( targetHeight - 3 ))|| (currentHeight > ( targetHeight + 3 ))))
         {
84         paintAsGreek( renderContext.getGraphics2D() );
           return;
86       }
         else
88       {
           PZoomHandler.setTargetRectangle( null );
90       }
       }
92     if ( zoomingNode instanceof edu.umd.cs.jazz.ZVisualLeaf )
       {
94       zoomingNode.editor().getTransformGroup().scale( scaleFactor );
       }
96     super.render(renderContext);
     }
98   /**
```

```
 * Insert the method's description here.
 * Creation date: (30-07-00 17:50:17)
 * @param newOriginalTransform java.awt.geom.AffineTransform
 */
public void setOriginalTransform(java.awt.geom.AffineTransform newOriginalTransform) {
   originalTransform = newOriginalTransform;
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 17:43:52)
 * @param newScaleFactor float
 */
public void setScaleFactor(float newScaleFactor) {
   scaleFactor = newScaleFactor;
}
/**
 * Insert the method's description here.
 * Creation date: (07-09-00 23:31:48)
 * @param newUpdating boolean
 */
public static void setUpdating(boolean newUpdating) {
   updating = newUpdating;
}
/**
 * Insert the method's description here.
 * Creation date: (30-07-00 17:26:06)
 * @param newZoomingNode edu.umd.cs.jazz.ZVisualLeaf
 */
public void setZoomingNode(edu.umd.cs.jazz.ZVisualLeaf newZoomingNode) {
   zoomingNode = newZoomingNode;
}
}
```

Listing C.24: PZText.java

```
package clientapp;

/**
 * Component for representing text items in the client panorama
 * Creation date: (16-06-00 23:05:35)
 * @author:
 */
public class PZText extends edu.umd.cs.jazz.component.ZText
{

/**
 * PZText constructor comment.
 */
public PZText() {
   super();
}
/**
 * PZText constructor comment.
 * @param str java.lang.String
 */
public PZText(String str) {
   super(str);
}
/**
 * PZText constructor comment.
 * @param str java.lang.String
 * @param font java.awt.Font
 */
public PZText(String str, java.awt.Font font) {
   super(str, font);
}
/**
```

```
 * Insert the method's description here.
 * Creation date: (16-06-00 23:07:51)
 * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
 */
public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
{
   //renderContext.setGreekText(true);
   super.render(renderContext);
}
}
```

Listing C.25: RemoveFiles.java

```
package clientapp;


 import java.awt.event.*;
 import javax.swing.border.*;
 import javax.swing.BorderFactory;
 import javax.swing.DefaultListModel;
 import java.io.File;

/**
 * This class implements the remove files dialog box.
 * Creation date: (22-06-00 23:00:55)
 * @author:
 */

public class RemoveFiles extends javax.swing.JDialog {

   public class RemoveButtonHandler implements ActionListener
   {
      public void actionPerformed(ActionEvent actionEvent)
      {
         submitAndClose();
      }
   }

   public class CancelButtonHandler implements ActionListener
   {
      public void actionPerformed(ActionEvent actionEvent)
      {
         dispose();
      }
   }

   public class MoveLeftButtonHandler implements ActionListener
   {
      public void actionPerformed(ActionEvent actionEvent)
      {
         moveFromDisplayedToRemove();
      }
   }

   public class MoveRightButtonHandler implements ActionListener
   {
      public void actionPerformed(ActionEvent actionEvent)
      {
         moveFromRemoveToDisplayed();
      }
   }
   private javax.swing.JButton ivjCancelButton = null;
   private javax.swing.JPanel ivjJDialogContentPane = null;
   private javax.swing.JScrollPane ivjJScrollPane1 = null;
   private javax.swing.JButton ivjRemoveButton = null;
   private javax.swing.DefaultListModel listModel = new DefaultListModel();
   private ClientApplication clientApplication;
```

```
    private javax.swing.JPanel ivjButtonPanel = null;
    private javax.swing.JLabel ivjDisplayedLabel = null;
    private javax.swing.JLabel ivjDocumentsToRemove = null;
    private javax.swing.JScrollPane ivjJScrollPane2 = null;
    private javax.swing.JList ivjRemoveList = null;
    private javax.swing.JList ivjDisplayedList = null;
    private javax.swing.JButton ivjMoveLeftButton = null;
    private javax.swing.JButton ivjMoveRightButton = null;
    private javax.swing.DefaultListModel removeListModel = new DefaultListModel();
    private javax.swing.JPanel ivjBottomPanel = null;
    private java.awt.FlowLayout ivjBottomPanelFlowLayout = null;
    private javax.swing.JPanel ivjMainPanel = null;
/**
 * RemoveFiles constructor comment.
 */
public RemoveFiles() {
    super();
    initialize();
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 */
public RemoveFiles(java.awt.Dialog owner) {
    super(owner);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public RemoveFiles(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public RemoveFiles(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public RemoveFiles(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 */
public RemoveFiles(java.awt.Frame owner) {
    super(owner);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public RemoveFiles(java.awt.Frame owner, String title) {
    super(owner, title);
}
```

```
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public RemoveFiles(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * RemoveFiles constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public RemoveFiles(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * Insert the method's description here.
 * Creation date: (22-06-00 23:31:46)
 * @param paths java.lang.String[]
 */
public void fillListBox(java.util.HashSet localFiles)
{
   java.util.Iterator iterator = localFiles.iterator();
   while ( iterator.hasNext() == true )
   {
      getListModel().addElement( (File) iterator.next() );
   }
}
/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getBottomPanel() {
   if (ivjBottomPanel == null) {
      try {
         ivjBottomPanel = new javax.swing.JPanel();
         ivjBottomPanel.setName("BottomPanel");
         ivjBottomPanel.setLayout(getBottomPanelFlowLayout());
         ivjBottomPanel.add(getRemoveButton());
         getBottomPanel().add(getCancelButton(), getCancelButton().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjBottomPanel;
}
/**
 * Return the BottomPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getBottomPanelFlowLayout() {
   java.awt.FlowLayout ivjBottomPanelFlowLayout = null;
   try {
      /* Create part */
      ivjBottomPanelFlowLayout = new java.awt.FlowLayout();
      ivjBottomPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
```

```
      };
188   return ivjBottomPanelFlowLayout;
    }
190 /**
     * Return the ButtonPanel property value.
192  * @return javax.swing.JPanel
     */
194 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getButtonPanel() {
196   if (ivjButtonPanel == null) {
        try {
198         ivjButtonPanel = new javax.swing.JPanel();
            ivjButtonPanel.setName("ButtonPanel");
200         ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
            ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
202         ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
            ivjButtonPanel.setMaximumSize(new java.awt.Dimension(31, 2147483647));
204
            java.awt.GridBagConstraints constraintsMoveLeftButton = new java.awt.GridBagConstraints();
206         constraintsMoveLeftButton.gridx = 0; constraintsMoveLeftButton.gridy = 0;
            getButtonPanel().add(getMoveLeftButton(), constraintsMoveLeftButton);
208
            java.awt.GridBagConstraints constraintsMoveRightButton = new java.awt.GridBagConstraints();
210         constraintsMoveRightButton.gridx = 0; constraintsMoveRightButton.gridy = 1;
            getButtonPanel().add(getMoveRightButton(), constraintsMoveRightButton);
212         // user code begin {1}
            // user code end
214     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
216         // user code end
            handleException(ivjExc);
218     }
      }
220   return ivjButtonPanel;
    }
222 /**
     * Return the CancelButton property value.
224  * @return javax.swing.JButton
     */
226 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getCancelButton() {
228   if (ivjCancelButton == null) {
        try {
230         ivjCancelButton = new javax.swing.JButton();
            ivjCancelButton.setName("CancelButton");
232         ivjCancelButton.setText("Cancel");
            // user code begin {1}
234         ivjCancelButton.addActionListener(new CancelButtonHandler() );
            // user code end
236     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
238         // user code end
            handleException(ivjExc);
240     }
      }
242   return ivjCancelButton;
    }
244 /**
     * Insert the method's description here.
246  * Creation date: (23-06-00 00:15:50)
     * @return clientapp.ClientApplication
248  */
    public ClientApplication getClientApplication() {
250   return clientApplication;
    }
252 /**
```

```
     * Return the DisplayedLabel property value.
254  * @return javax.swing.JLabel
     */
256  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JLabel getDisplayedLabel() {
258     if (ivjDisplayedLabel == null) {
           try {
260           ivjDisplayedLabel = new javax.swing.JLabel();
              ivjDisplayedLabel.setName("DisplayedLabel");
262           ivjDisplayedLabel.setText("Currently␣displayed");
              // user code begin {1}
264           // user code end
           } catch (java.lang.Throwable ivjExc) {
266           // user code begin {2}
              // user code end
268           handleException(ivjExc);
           }
270     }
        return ivjDisplayedLabel;
272  }
     /**
274  * Return the JList1 property value.
     * @return javax.swing.JList
276  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
278  private javax.swing.JList getDisplayedList() {
        if (ivjDisplayedList == null) {
280        try {
              ivjDisplayedList = new javax.swing.JList();
282           ivjDisplayedList.setName("DisplayedList");
              ivjDisplayedList.setBounds(0, 0, 0, 0);
284           // user code begin {1}
              ivjDisplayedList.setModel( getListModel() );
286           // user code end
           } catch (java.lang.Throwable ivjExc) {
288           // user code begin {2}
              // user code end
290           handleException(ivjExc);
           }
292     }
        return ivjDisplayedList;
294  }
     /**
296  * Return the DocumentsToRemove property value.
     * @return javax.swing.JLabel
298  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
300  private javax.swing.JLabel getDocumentsToRemove() {
        if (ivjDocumentsToRemove == null) {
302        try {
              ivjDocumentsToRemove = new javax.swing.JLabel();
304           ivjDocumentsToRemove.setName("DocumentsToRemove");
              ivjDocumentsToRemove.setText("To␣be␣removed");
306           // user code begin {1}
              // user code end
308        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
310           // user code end
              handleException(ivjExc);
312        }
        }
314     return ivjDocumentsToRemove;
     }
316  /**
     * Return the JDialogContentPane property value.
318  * @return javax.swing.JPanel
```

```
     */
320  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJDialogContentPane() {
322     if (ivjJDialogContentPane == null) {
           try {
324           ivjJDialogContentPane = new javax.swing.JPanel();
              ivjJDialogContentPane.setName("JDialogContentPane");
326           ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
              getJDialogContentPane().add(getBottomPanel(), "South");
328           getJDialogContentPane().add(getMainPanel(), "Center");
              // user code begin {1}
330           // user code end
           } catch (java.lang.Throwable ivjExc) {
332           // user code begin {2}
              // user code end
334           handleException(ivjExc);
           }
336     }
        return ivjJDialogContentPane;
338  }
     /**
340   * Return the JScrollPane1 property value.
      * @return javax.swing.JScrollPane
342   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
344  private javax.swing.JScrollPane getJScrollPane1() {
        if (ivjJScrollPane1 == null) {
346        try {
              ivjJScrollPane1 = new javax.swing.JScrollPane();
348           ivjJScrollPane1.setName("JScrollPane1");
              getJScrollPane1().setViewportView(getDisplayedList());
350           // user code begin {1}
              // user code end
352        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
354           // user code end
              handleException(ivjExc);
356        }
        }
358     return ivjJScrollPane1;
     }
360  /**
      * Return the JScrollPane2 property value.
362   * @return javax.swing.JScrollPane
      */
364  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JScrollPane getJScrollPane2() {
366     if (ivjJScrollPane2 == null) {
           try {
368           ivjJScrollPane2 = new javax.swing.JScrollPane();
              ivjJScrollPane2.setName("JScrollPane2");
370           getJScrollPane2().setViewportView(getRemoveList());
              // user code begin {1}
372           // user code end
           } catch (java.lang.Throwable ivjExc) {
374           // user code begin {2}
              // user code end
376           handleException(ivjExc);
           }
378     }
        return ivjJScrollPane2;
380  }
     /**
382   * Insert the method's description here.
      * Creation date: (22-06-00 23:40:02)
384   * @return javax.swing.DefaultListModel
```

```
 */
public javax.swing.DefaultListModel getListModel() {
   return listModel;
}
/**
 * Return the JPanel2 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getMainPanel() {
   if (ivjMainPanel == null) {
      try {
         ivjMainPanel = new javax.swing.JPanel();
         ivjMainPanel.setName("MainPanel");
         ivjMainPanel.setLayout(new java.awt.GridBagLayout());

         java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
         constraintsButtonPanel.gridx = 1; constraintsButtonPanel.gridy = 1;
         constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
         getMainPanel().add(getButtonPanel(), constraintsButtonPanel);

         java.awt.GridBagConstraints constraintsDisplayedLabel = new java.awt.GridBagConstraints();
         constraintsDisplayedLabel.gridx = 0; constraintsDisplayedLabel.gridy = 0;
         constraintsDisplayedLabel.anchor = java.awt.GridBagConstraints.WEST;
         constraintsDisplayedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getDisplayedLabel(), constraintsDisplayedLabel);

         java.awt.GridBagConstraints constraintsJScrollPane1 = new java.awt.GridBagConstraints();
         constraintsJScrollPane1.gridx = 0; constraintsJScrollPane1.gridy = 1;
         constraintsJScrollPane1.fill = java.awt.GridBagConstraints.BOTH;
         constraintsJScrollPane1.anchor = java.awt.GridBagConstraints.WEST;
         constraintsJScrollPane1.weightx = 1.0;
         constraintsJScrollPane1.weighty = 1.0;
         constraintsJScrollPane1.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getJScrollPane1(), constraintsJScrollPane1);

         java.awt.GridBagConstraints constraintsJScrollPane2 = new java.awt.GridBagConstraints();
         constraintsJScrollPane2.gridx = 2; constraintsJScrollPane2.gridy = 1;
         constraintsJScrollPane2.fill = java.awt.GridBagConstraints.BOTH;
         constraintsJScrollPane2.anchor = java.awt.GridBagConstraints.EAST;
         constraintsJScrollPane2.weightx = 1.0;
         constraintsJScrollPane2.weighty = 1.0;
         constraintsJScrollPane2.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getJScrollPane2(), constraintsJScrollPane2);

         java.awt.GridBagConstraints constraintsDocumentsToRemove = new java.awt.GridBagConstraints();
         constraintsDocumentsToRemove.gridx = 2; constraintsDocumentsToRemove.gridy = 0;
         constraintsDocumentsToRemove.anchor = java.awt.GridBagConstraints.WEST;
         constraintsDocumentsToRemove.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getDocumentsToRemove(), constraintsDocumentsToRemove);
         // user code begin {1}
         getMainPanel().setBorder( ClientConstants.createDialogBorder( getMainPanel().getBackground() ));
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjMainPanel;
}
/**
 * Return the AddButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
    private javax.swing.JButton getMoveLeftButton() {
452    if (ivjMoveLeftButton == null) {
       try {
454       ivjMoveLeftButton = new javax.swing.JButton();
          ivjMoveLeftButton.setName("MoveLeftButton");
456       ivjMoveLeftButton.setToolTipText("Add␣to␣list");
          ivjMoveLeftButton.setMnemonic('a');
458       ivjMoveLeftButton.setText("");
          ivjMoveLeftButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
              toolbarButtonGraphics/navigation/Forward24.gif")));
460       ivjMoveLeftButton.setBorderPainted(false);
          ivjMoveLeftButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
462       // user code begin {1}
          ivjMoveLeftButton.addActionListener(new MoveLeftButtonHandler());
464       // user code end
       } catch (java.lang.Throwable ivjExc) {
466       // user code begin {2}
          // user code end
468       handleException(ivjExc);
       }
470    }
       return ivjMoveLeftButton;
472 }
    /**
474  * Return the RemoveButton1 property value.
     * @return javax.swing.JButton
476  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
478 private javax.swing.JButton getMoveRightButton() {
       if (ivjMoveRightButton == null) {
480    try {
          ivjMoveRightButton = new javax.swing.JButton();
482       ivjMoveRightButton.setName("MoveRightButton");
          ivjMoveRightButton.setToolTipText("Remove␣from␣list");
484       ivjMoveRightButton.setMnemonic('r');
          ivjMoveRightButton.setText("");
486       ivjMoveRightButton.setMaximumSize(new java.awt.Dimension(29, 35));
          ivjMoveRightButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
              toolbarButtonGraphics/navigation/Back24.gif")));
488       ivjMoveRightButton.setBorderPainted(false);
          ivjMoveRightButton.setPreferredSize(new java.awt.Dimension(29, 35));
490       ivjMoveRightButton.setMinimumSize(new java.awt.Dimension(29, 35));
          ivjMoveRightButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
492       // user code begin {1}
          ivjMoveRightButton.addActionListener(new MoveRightButtonHandler());
494       // user code end
       } catch (java.lang.Throwable ivjExc) {
496       // user code begin {2}
          // user code end
498       handleException(ivjExc);
       }
500    }
       return ivjMoveRightButton;
502 }
    /**
504  * Return the RemoveButton property value.
     * @return javax.swing.JButton
506  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
508 private javax.swing.JButton getRemoveButton() {
       if (ivjRemoveButton == null) {
510    try {
          ivjRemoveButton = new javax.swing.JButton();
512       ivjRemoveButton.setName("RemoveButton");
          ivjRemoveButton.setMnemonic('r');
514       ivjRemoveButton.setText("Remove");
```

```
            // user code begin {1}
516         ivjRemoveButton.addActionListener(new RemoveButtonHandler() );
            // user code end
518      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
520         // user code end
            handleException(ivjExc);
522      }
         }
524      return ivjRemoveButton;
      }
526   /**
       * Return the RemoveList property value.
528    * @return javax.swing.JList
       */
530   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JList getRemoveList() {
532      if (ivjRemoveList == null) {
            try {
534         ivjRemoveList = new javax.swing.JList();
            ivjRemoveList.setName("RemoveList");
536         ivjRemoveList.setBackground(java.awt.Color.white);
            ivjRemoveList.setBounds(0, 0, 160, 120);
538         // user code begin {1}
            ivjRemoveList.setModel(removeListModel);
540         // user code end
            } catch (java.lang.Throwable ivjExc) {
542         // user code begin {2}
            // user code end
544         handleException(ivjExc);
            }
546      }
         return ivjRemoveList;
548   }
      /**
550    * Insert the method's description here.
       * Creation date: (23-06-00 19:55:23)
552    * @return javax.swing.DefaultListModel
       */
554   public javax.swing.DefaultListModel getRemoveListModel() {
         return removeListModel;
556   }
      /**
558    * Called whenever the part throws an exception.
       * @param exception java.lang.Throwable
560    */
      private void handleException(java.lang.Throwable exception) {
562
         /* Uncomment the following lines to print uncaught exceptions to stdout */
564      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
         // exception.printStackTrace(System.out);
566   }
      /**
568    * Insert the method's description here.
       * Creation date: (22-06-00 23:11:53)
570    */
      public void initialize() {
572      try {
            // user code begin {1}
574         // user code end
            setName("RemoveFiles");
576         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            setSize(600, 300);
578         setTitle("Remove␣documents");
            setContentPane(getJDialogContentPane());
580      } catch (java.lang.Throwable ivjExc) {
```

```
      handleException(ivjExc);
   }
   // user code begin {2}
   getJDialogContentPane().setBorder( javax.swing.BorderFactory.createMatteBorder(6,6,6,6,
        getJDialogContentPane().getBackground() ));
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      RemoveFiles aRemoveFiles;
      aRemoveFiles = new RemoveFiles();
      aRemoveFiles.setModal(true);
      aRemoveFiles.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aRemoveFiles.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 20:04:34)
 */
public void moveFromDisplayedToRemove()
{
   for (int i = listModel.size()-1; i >= 0; i--)
   {
      if ( getDisplayedList().isSelectedIndex(i) )
      {
         removeListModel.addElement( listModel.get(i) );
         listModel.remove(i);
      }
   }
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 20:05:56)
 */
public void moveFromRemoveToDisplayed()
{
   for (int i = removeListModel.size()-1; i >= 0; i--)
   {
      if ( getRemoveList().isSelectedIndex(i) )
      {
         listModel.addElement( removeListModel.get(i) );
         removeListModel.remove(i);
      }
   }
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 00:15:50)
 * @param newClientApplication clientapp.ClientApplication
 */
public void setClientApplication(ClientApplication newClientApplication) {
   clientApplication = newClientApplication;
}
/**
```

```
 * Insert the method's description here.
 * Creation date: (22-06-00 23:40:02)
 * @param newListModel javax.swing.DefaultListModel
 */
public void setListModel(javax.swing.DefaultListModel newListModel) {
   listModel = newListModel;
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 19:55:23)
 * @param newRemoveListModel javax.swing.DefaultListModel
 */
public void setRemoveListModel(javax.swing.DefaultListModel newRemoveListModel) {
   removeListModel = newRemoveListModel;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 02:38:38)
 */
public void show()
{
   listModel.clear();
   removeListModel.clear();
   fillListBox( clientApplication.getClientManager().getClientInfo().getLocalFiles() );
   // default positioning puts the dialog at center of screen. The method used is a slight hack, but
        works fine.
   setLocationRelativeTo( this );
   setLocation( (int) getLocation().getX(), (int) getLocation().getY() );
   super.show();
}
/**
 * Insert the method's description here.
 * Creation date: (23-06-00 00:12:51)
 */
public void submitAndClose()
{
   if ( removeListModel.size() > 0)
   {
      File removables[] = new File[removeListModel.size()];
      File remainingFiles[] = new File[listModel.size()];
      for (int i = 0; i < removables.length; i++)
      {
         removables[i] = (File)removeListModel.get(i);
      }
      clientApplication.getClientManager().removeDocuments( removables );
   }
   else
   {
      clientApplication.initMessageArea( 1, 1, 1, ClientConstants.getNOTHING_TO_REMOVE_MESSAGE() );
   }
   dispose();
}
}
```

Listing C.26: Validator.java

```
package clientapp;

/**
 * Factory class with various static member functions for validating data entered into the client
      application.
 * Creation date: (26-07-00 15:44:56)
 * @author:
 */
public class Validator
{
   private ClientApplication ClientApplication = null;
```

```
/**
 * Validator constructor comment.
 */
public Validator( ClientApplication clientApplication )
{
   super();
   setClientApplication( clientApplication );
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 16:07:18)
 * @return clientapp.ClientApplication
 */
public ClientApplication getClientApplication() {
   return ClientApplication;
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 16:07:18)
 * @param newClientApplication clientapp.ClientApplication
 */
public void setClientApplication(ClientApplication newClientApplication) {
   ClientApplication = newClientApplication;
}
}
```

Listing C.27: GroupManagementServer.java

```
package serverapp;

import com.sun.media.jsdt.*;
/**
 * This class is intended for use by PeerView servers connecting to a JSDT session.
 * Creation date: (03-07-00 12:17:46)
 * @author:
 */
public class GroupManagementServer implements Client, ChannelConsumer {
   private java.lang.String name = "";
/**
 * GroupManagementServer constructor comment.
 */
public GroupManagementServer() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:30:58)
 * @param ai com.sun.media.jsdt.AuthenticationInfo
 */
public Object authenticate(AuthenticationInfo ai)
{
   return null;
}
/**
 * dataReceived method comment.
 */
public synchronized void dataReceived(Data data)
{


}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:21:42)
 * @return java.lang.String
 */
public java.lang.String getName() {
```

```
40      return name;
    }
42  /**
     * Insert the method's description here.
44   * Creation date: (03-07-00 12:21:42)
     * @param newName java.lang.String
46   */
    public void setName(java.lang.String newName) {
48      name = newName;
    }
50  }
```

Listing C.28: PeerViewServer.java

```
    package serverapp;
 2
    import com.sun.media.jsdt.*;
 4  /**
     * The PeerView server class.
 6   * Creation date: (30-06-00 10:00:51)
     * @author:
 8   */
    public class PeerViewServer extends ServerInfo implements Client, ChannelConsumer {
10      private java.lang.String name;
    /**
12   * PeerViewServer constructor comment.
     */
14  public PeerViewServer(String nameArg)
    {
16      super();
        name = nameArg;
18  }
    /**
20   * authenticate method comment.
     */
22  public Object authenticate(AuthenticationInfo arg1) {
        return null;
24  }
    /**
26   * Insert the method's description here.
     * Creation date: (30-06-00 14:24:04)
28   * @param data com.sun.media.jsdt.Data
     */
30  public synchronized void dataReceived(Data data)
    {
32      // System.out.println( data.getDataAsString() );
    }
34  /**
     * Insert the method's description here.
36   * Creation date: (30-06-00 10:15:32)
     * @return java.lang.String
38   */
    public java.lang.String getName() {
40      return name;
    }
42  /**
     * Insert the method's description here.
44   * Creation date: (30-06-00 10:15:32)
     * @param newName java.lang.String
46   */
    public void setName(java.lang.String newName) {
48      name = newName;
    }
50  }
```

Listing C.29: ServerApplication.java

```
package serverapp;

 import com.sun.media.jsdt.*;
 import peerviewmisc.*;
 import java.util.*;
 import java.awt.Dimension;
 import java.awt.Point;

 /**
 * This class implements the server application main window.
 * Creation date: (29-06-00 17:57:51)
 * @author:
 */
public class ServerApplication extends javax.swing.JFrame {
   public Channel channel;
   private peerviewmisc.AboutDialog ivjAboutDialog = null;
   private javax.swing.JMenuItem ivjAboutMenuItem = null;
   private javax.swing.JMenuItem ivjContentsMenuItem = null;
   IvjEventHandler ivjEventHandler = new IvjEventHandler();
   private javax.swing.JButton ivjExitButton = null;
   private javax.swing.JMenuItem ivjExitMenuItem = null;
   private javax.swing.JButton ivjHelpButton = null;
   private javax.swing.JMenu ivjHelpMenu = null;
   private javax.swing.JPanel ivjJFrameContentPane = null;
   private javax.swing.JPanel ivjJPanel1 = null;
   private javax.swing.JScrollPane ivjJScrollPane1 = null;
   private javax.swing.JSeparator ivjJSeparator1 = null;
   private javax.swing.JToolBar ivjJToolBar1 = null;
   private javax.swing.JTable ivjScrollPaneTable = null;
   private javax.swing.JMenuBar ivjServerApplicationJMenuBar = null;
   private javax.swing.JMenu ivjServerMenu = null;
   private ServerSetup ivjServerSetup = null;
   private javax.swing.JButton ivjSetupButton = null;
   private javax.swing.JMenuItem ivjSetupMenuItem = null;
   private ServerManager serverManager = null;
   private javax.swing.table.DefaultTableModel tableModel = null;
   private HelpDialog ivjHelpDialog1 = null;

class IvjEventHandler implements java.awt.event.ActionListener {
      public void actionPerformed(java.awt.event.ActionEvent e) {
         if (e.getSource() == ServerApplication.this.getSetupMenuItem())
            connEtoM1(e);
         if (e.getSource() == ServerApplication.this.getAboutMenuItem())
            connEtoM3(e);
         if (e.getSource() == ServerApplication.this.getExitButton())
            connEtoM4(e);
         if (e.getSource() == ServerApplication.this.getHelpButton())
            connEtoM5(e);
         if (e.getSource() == ServerApplication.this.getContentsMenuItem())
            connEtoM6(e);
         if (e.getSource() == ServerApplication.this.getSetupButton())
            connEtoM7(e);
         if (e.getSource() == ServerApplication.this.getExitMenuItem())
            connEtoM2(e);
         if (e.getSource() == ServerApplication.this.getExitButton())
            connEtoM8(e);
      };
   };
/**
 * ServerApplication constructor comment.
 */
public ServerApplication() {
   super();
   initialize();
}
/**
```

```
     * ServerApplication constructor comment.
68   * @param title java.lang.String
     */
70  public ServerApplication(String title) {
       super(title);
72  }
    /**
74   * Center and size the window at predefined dimensions and position relative to the screen resolution
     * Creation date: (05-08-00 19:20:24)
76   */
    public void center()
78  {
       Dimension size;
80     Dimension preferredSize = new Dimension(
                    Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
                         getFRAME_HORIZONTAL_SIZE() )),
82                  Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
                         getFRAME_VERTICAL_SIZE() )));
       Dimension defaultSize = new Dimension(
84                  Integer.parseInt( ServerConstants.getDEFAULT_FRAME_HORIZONTAL_SIZE() ),
                    Integer.parseInt( ServerConstants.getDEFAULT_FRAME_VERTICAL_SIZE() ));
86     Point position;
       Point preferredPosition = new Point(
88                  Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
                         getFRAME_X_COORDINATE() )),
                    Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
                         getFRAME_Y_COORDINATE() )));
90     Point defaultPosition = new Point(
                    Integer.parseInt( ServerConstants.getDEFAULT_FRAME_X_COORDINATE() ),
92                  Integer.parseInt( ServerConstants.getDEFAULT_FRAME_Y_COORDINATE() ));
       // if no value has overridden the default setting then query the OS for the dimensions of the display
94     if ( preferredSize.equals( defaultSize ))
       {
96        size = new Dimension(  (int) (getToolkit().getScreenSize().getWidth() * ServerConstants.
              getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
                         (int) (getToolkit().getScreenSize().getHeight() * ServerConstants.
                              getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) );
98     }
       else
100    {
          size = preferredSize;
102    }
       if ( preferredPosition.equals( defaultPosition ))
104    {
          position = new Point(  (int)( getToolkit().getScreenSize().getWidth() * ( (1.0 - ServerConstants.
              getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2) ),
106                      (int)( getToolkit().getScreenSize().getHeight() * ( (1.0 - ServerConstants.
                              getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2) ));
       }
108    else
       {
110       position = preferredPosition;
       }
112    // System.out.println( size.toString() );
       setSize( size );
114    setLocation( position );
       setState( java.awt.Frame.NORMAL );
116    repaint();
    }
118 /**
     * connEtoM1: (SetupMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> ServerSetup.show()V)
120  * @param arg1 java.awt.event.ActionEvent
     */
122 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM1(java.awt.event.ActionEvent arg1) {
124    try {
```

```
        // user code begin {1}
126     // user code end
        getServerSetup().show();
128     // user code begin {2}
        // user code end
130  } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
132     // user code end
        handleException(ivjExc);
134  }
     }
136  /**
      * connEtoM2: (ExitMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.
          dispose()V)
138   * @param arg1 java.awt.event.ActionEvent
      */
140  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM2(java.awt.event.ActionEvent arg1) {
142     try {
        // user code begin {1}
144     // user code end
        this.terminateAndClose();
146     // user code begin {2}
        // user code end
148  } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
150     // user code end
        handleException(ivjExc);
152  }
     }
154  /**
      * connEtoM3: (AboutMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> AboutDialog.show()V)
156   * @param arg1 java.awt.event.ActionEvent
      */
158  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM3(java.awt.event.ActionEvent arg1) {
160     try {
        // user code begin {1}
162     // user code end
        getAboutDialog().show();
164     // user code begin {2}
        // user code end
166  } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
168     // user code end
        handleException(ivjExc);
170  }
     }
172  /**
      * connEtoM4: (NewGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.show()V)
174   * @param arg1 java.awt.event.ActionEvent
      */
176  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM4(java.awt.event.ActionEvent arg1) {
178     try {
        // user code begin {1}
180     // user code end
        this.dispose();
182     // user code begin {2}
        // user code end
184  } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
186     // user code end
        handleException(ivjExc);
188  }
     }
```

```
190  /**
      * connEtoM5: (EditGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> EditGroup1.show()V)
192   * @param arg1 java.awt.event.ActionEvent
      */
194  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM5(java.awt.event.ActionEvent arg1) {
196     try {
           // user code begin {1}
198        // user code end
           getHelpDialog1().show();
200        // user code begin {2}
           // user code end
202     } catch (java.lang.Throwable ivjExc) {
           // user code begin {3}
204        // user code end
           handleException(ivjExc);
206     }
     }
208  /**
      * connEtoM6: (DeleteGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> DeleteGroup1.show
          ()V)
210   * @param arg1 java.awt.event.ActionEvent
      */
212  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM6(java.awt.event.ActionEvent arg1) {
214     try {
           // user code begin {1}
216        // user code end
           getHelpDialog1().show();
218        // user code begin {2}
           // user code end
220     } catch (java.lang.Throwable ivjExc) {
           // user code begin {3}
222        // user code end
           handleException(ivjExc);
224     }
     }
226  /**
      * connEtoM7: (DefaultToolBarButton.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.
          show()V)
228   * @param arg1 java.awt.event.ActionEvent
      */
230  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM7(java.awt.event.ActionEvent arg1) {
232     try {
           // user code begin {1}
234        // user code end
           getServerSetup().show();
236        // user code begin {2}
           // user code end
238     } catch (java.lang.Throwable ivjExc) {
           // user code begin {3}
240        // user code end
           handleException(ivjExc);
242     }
     }
244  /**
      * connEtoM8: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.
          terminateAndClose()V)
246   * @param arg1 java.awt.event.ActionEvent
      */
248  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void connEtoM8(java.awt.event.ActionEvent arg1) {
250     try {
           // user code begin {1}
252        // user code end
```

```
         this.terminateAndClose();
254      // user code begin {2}
         // user code end
256   } catch (java.lang.Throwable ivjExc) {
         // user code begin {3}
258      // user code end
         handleException(ivjExc);
260   }
   }
262 /**
    * Insert the method's description here.
264 * Creation date: (07-07-00 01:53:28)
    */
266 protected void finalize() throws Throwable
   {
268   terminateAndClose();
      super.finalize();
270 }
   /**
272 * Return the AboutDialog property value.
    * @return clientapp.AboutDialog
274 */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
276 private peerviewmisc.AboutDialog getAboutDialog() {
      if (ivjAboutDialog == null) {
278      try {
            ivjAboutDialog = new peerviewmisc.AboutDialog();
280         ivjAboutDialog.setName("AboutDialog");
            ivjAboutDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
282         // user code begin {1}
            // user code end
284      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
286         // user code end
            handleException(ivjExc);
288      }
      }
290   return ivjAboutDialog;
   }
292 /**
    * Return the AboutMenuItem property value.
294 * @return javax.swing.JMenuItem
    */
296 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getAboutMenuItem() {
298   if (ivjAboutMenuItem == null) {
         try {
300         ivjAboutMenuItem = new javax.swing.JMenuItem();
            ivjAboutMenuItem.setName("AboutMenuItem");
302         ivjAboutMenuItem.setText("About");
            // user code begin {1}
304         // user code end
         } catch (java.lang.Throwable ivjExc) {
306         // user code begin {2}
            // user code end
308         handleException(ivjExc);
         }
310   }
      return ivjAboutMenuItem;
312 }
   /**
314 * Return the ContentsMenuItem property value.
    * @return javax.swing.JMenuItem
316 */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
318 private javax.swing.JMenuItem getContentsMenuItem() {
```

```
      if (ivjContentsMenuItem == null) {
320      try {
            ivjContentsMenuItem = new javax.swing.JMenuItem();
322         ivjContentsMenuItem.setName("ContentsMenuItem");
            ivjContentsMenuItem.setText("Contents");
324         // user code begin {1}
            // user code end
326      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
328         // user code end
            handleException(ivjExc);
330      }
      }
332   return ivjContentsMenuItem;
   }
334 /**
    * Return the ExitButton property value.
336 * @return javax.swing.JButton
    */
338 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JButton getExitButton() {
340   if (ivjExitButton == null) {
      try {
342      ivjExitButton = new javax.swing.JButton();
         ivjExitButton.setName("ExitButton");
344      ivjExitButton.setToolTipText("Exit");
         ivjExitButton.setText("");
346      ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
         ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
348      ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
             general/Stop24.gif")));
         ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
350      // user code begin {1}
         // user code end
352   } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
354      // user code end
         handleException(ivjExc);
356   }
      }
358   return ivjExitButton;
   }
360 /**
    * Return the ExitMenuItem property value.
362 * @return javax.swing.JMenuItem
    */
364 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getExitMenuItem() {
366   if (ivjExitMenuItem == null) {
      try {
368      ivjExitMenuItem = new javax.swing.JMenuItem();
         ivjExitMenuItem.setName("ExitMenuItem");
370      ivjExitMenuItem.setText("Exit");
         // user code begin {1}
372      // user code end
      } catch (java.lang.Throwable ivjExc) {
374      // user code begin {2}
         // user code end
376      handleException(ivjExc);
      }
378   }
      return ivjExitMenuItem;
380 }
   /**
382 * Return the HelpButton property value.
    * @return javax.swing.JButton
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getHelpButton() {
   if (ivjHelpButton == null) {
      try {
         ivjHelpButton = new javax.swing.JButton();
         ivjHelpButton.setName("HelpButton");
         ivjHelpButton.setToolTipText("Help");
         ivjHelpButton.setText("");
         ivjHelpButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
         ivjHelpButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
         ivjHelpButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
             general/Help24.gif")));
         ivjHelpButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjHelpButton;
}
/**
 * Return the HelpDialog1 property value.
 * @return peerviewmisc.HelpDialog
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private peerviewmisc.HelpDialog getHelpDialog1() {
   if (ivjHelpDialog1 == null) {
      try {
         ivjHelpDialog1 = new peerviewmisc.HelpDialog();
         ivjHelpDialog1.setName("HelpDialog1");
         ivjHelpDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjHelpDialog1;
}
/**
 * Return the HelpMenu property value.
 * @return javax.swing.JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenu getHelpMenu() {
   if (ivjHelpMenu == null) {
      try {
         ivjHelpMenu = new javax.swing.JMenu();
         ivjHelpMenu.setName("HelpMenu");
         ivjHelpMenu.setText("Help");
         ivjHelpMenu.add(getContentsMenuItem());
         ivjHelpMenu.add(getAboutMenuItem());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
```

```
      return ivjHelpMenu;
   }
   /**
    * Return the JFrameContentPane property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJFrameContentPane() {
      if (ivjJFrameContentPane == null) {
         try {
            ivjJFrameContentPane = new javax.swing.JPanel();
            ivjJFrameContentPane.setName("JFrameContentPane");
            ivjJFrameContentPane.setLayout(new java.awt.BorderLayout());
            getJFrameContentPane().add(getJToolBar1(), "North");
            getJFrameContentPane().add(getJPanel1(), "Center");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJFrameContentPane;
   }
   /**
    * Return the JPanel1 property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJPanel1() {
      if (ivjJPanel1 == null) {
         try {
            ivjJPanel1 = new javax.swing.JPanel();
            ivjJPanel1.setName("JPanel1");
            ivjJPanel1.setLayout(new java.awt.BorderLayout());
            getJPanel1().add(getJScrollPane1(), "Center");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJPanel1;
   }
   /**
    * Return the JScrollPane1 property value.
    * @return javax.swing.JScrollPane
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JScrollPane getJScrollPane1() {
      if (ivjJScrollPane1 == null) {
         try {
            ivjJScrollPane1 = new javax.swing.JScrollPane();
            ivjJScrollPane1.setName("JScrollPane1");
            getJScrollPane1().setViewportView(getScrollPaneTable());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
```

```
      return ivjJScrollPane1;
   }
   /**
    * Return the JSeparator1 property value.
    * @return javax.swing.JSeparator
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JSeparator getJSeparator1() {
      if (ivjJSeparator1 == null) {
         try {
            ivjJSeparator1 = new javax.swing.JSeparator();
            ivjJSeparator1.setName("JSeparator1");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJSeparator1;
   }
   /**
    * Return the JToolBar1 property value.
    * @return javax.swing.JToolBar
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JToolBar getJToolBar1() {
      if (ivjJToolBar1 == null) {
         try {
            ivjJToolBar1 = new javax.swing.JToolBar();
            ivjJToolBar1.setName("JToolBar1");
            getJToolBar1().add(getSetupButton(), getSetupButton().getName());
            ivjJToolBar1.addSeparator();
            ivjJToolBar1.addSeparator();
            getJToolBar1().add(getHelpButton(), getHelpButton().getName());
            getJToolBar1().add(getExitButton(), getExitButton().getName());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJToolBar1;
   }
   /**
    * Return the ScrollPaneTable property value.
    * @return javax.swing.JTable
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTable getScrollPaneTable() {
      if (ivjScrollPaneTable == null) {
         try {
            ivjScrollPaneTable = new javax.swing.JTable();
            ivjScrollPaneTable.setName("ScrollPaneTable");
            getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
            getJScrollPane1().getViewport().setBackingStoreEnabled(true);
            ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
            ivjScrollPaneTable.setPreferredSize(new java.awt.Dimension(716,412));
            ivjScrollPaneTable.setBounds(0, 0, 716, 412);
            ivjScrollPaneTable.setPreferredScrollableViewportSize(new java.awt.Dimension(700, 400));
            ivjScrollPaneTable.setAutoCreateColumnsFromModel(true);
            // user code begin {1}
            ivjScrollPaneTable.setModel( getTableModel() );
```

```
            // user code end
582       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
584         // user code end
            handleException(ivjExc);
586       }
        }
588     return ivjScrollPaneTable;
    }
590 /**
     * Return the ServerApplicationJMenuBar property value.
592  * @return javax.swing.JMenuBar
     */
594 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenuBar getServerApplicationJMenuBar() {
596     if (ivjServerApplicationJMenuBar == null) {
        try {
598         ivjServerApplicationJMenuBar = new javax.swing.JMenuBar();
            ivjServerApplicationJMenuBar.setName("ServerApplicationJMenuBar");
600         ivjServerApplicationJMenuBar.add(getServerMenu());
            ivjServerApplicationJMenuBar.add(getHelpMenu());
602         // user code begin {1}
            // user code end
604     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
606         // user code end
            handleException(ivjExc);
608     }
        }
610     return ivjServerApplicationJMenuBar;
    }
612 /**
     * Insert the method's description here.
614  * Creation date: (29-06-00 18:47:13)
     * @return serverapp.ServerManager
616  */
    public ServerManager getServerManager()
618 {
        if ( serverManager == null )
620     {
            serverManager = new ServerManager();
622         serverManager.setServerApplication( this );
        }
624     return serverManager;
    }
626 /**
     * Return the ServerMenu property value.
628  * @return javax.swing.JMenu
     */
630 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenu getServerMenu() {
632     if (ivjServerMenu == null) {
        try {
634         ivjServerMenu = new javax.swing.JMenu();
            ivjServerMenu.setName("ServerMenu");
636         ivjServerMenu.setText("Setup");
            ivjServerMenu.add(getSetupMenuItem());
638         ivjServerMenu.add(getJSeparator1());
            ivjServerMenu.add(getExitMenuItem());
640         // user code begin {1}
            // user code end
642     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
644         // user code end
            handleException(ivjExc);
646     }
```

```
      }
648   return ivjServerMenu;
    }
650 /**
     * Return the ServerSetup property value.
652  * @return serverapp.ServerSetup
     */
654 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private ServerSetup getServerSetup() {
656   if (ivjServerSetup == null) {
        try {
658        ivjServerSetup = new serverapp.ServerSetup();
           ivjServerSetup.setName("ServerSetup");
660        ivjServerSetup.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           // user code begin {1}
662        // user code end
        } catch (java.lang.Throwable ivjExc) {
664        // user code begin {2}
           // user code end
666        handleException(ivjExc);
        }
668   }
      return ivjServerSetup;
670 }
    /**
672  * Return the SetupButton property value.
     * @return javax.swing.JButton
674  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
676 private javax.swing.JButton getSetupButton() {
      if (ivjSetupButton == null) {
678     try {
           ivjSetupButton = new javax.swing.JButton();
680        ivjSetupButton.setName("SetupButton");
           ivjSetupButton.setToolTipText("Server␣setup");
682        ivjSetupButton.setText("");
           ivjSetupButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
684        ivjSetupButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
           ivjSetupButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
               general/Properties24.gif")));
686        ivjSetupButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
           // user code begin {1}
688        // user code end
        } catch (java.lang.Throwable ivjExc) {
690        // user code begin {2}
           // user code end
692        handleException(ivjExc);
        }
694   }
      return ivjSetupButton;
696 }
    /**
698  * Return the SetupMenuItem property value.
     * @return javax.swing.JMenuItem
700  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
702 private javax.swing.JMenuItem getSetupMenuItem() {
      if (ivjSetupMenuItem == null) {
704     try {
           ivjSetupMenuItem = new javax.swing.JMenuItem();
706        ivjSetupMenuItem.setName("SetupMenuItem");
           ivjSetupMenuItem.setText("Setup");
708        // user code begin {1}
           // user code end
710     } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
```

```
712         // user code end
            handleException(ivjExc);
714      }
      }
716   return ivjSetupMenuItem;
   }
718 /**
     * Insert the method's description here.
720  * Creation date: (17-07-00 21:34:13)
     * @return javax.swing.table.DefaultTableModel
722  */
   public javax.swing.table.DefaultTableModel getTableModel() {
724   if ( tableModel == null )
      {
726      tableModel = new javax.swing.table.DefaultTableModel();
      }
728   return tableModel;
   }
730 /**
     * Called whenever the part throws an exception.
732  * @param exception java.lang.Throwable
     */
734 private void handleException(java.lang.Throwable exception) {

736   /* Uncomment the following lines to print uncaught exceptions to stdout */
      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
738   // exception.printStackTrace(System.out);
   }
740 /**
     * Initializes connections
742  * @exception java.lang.Exception The exception description.
     */
744 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private void initConnections() throws java.lang.Exception {
746   // user code begin {1}
      // user code end
748   getSetupMenuItem().addActionListener(ivjEventHandler);
      getAboutMenuItem().addActionListener(ivjEventHandler);
750   getExitButton().addActionListener(ivjEventHandler);
      getHelpButton().addActionListener(ivjEventHandler);
752   getContentsMenuItem().addActionListener(ivjEventHandler);
      getSetupButton().addActionListener(ivjEventHandler);
754   getExitMenuItem().addActionListener(ivjEventHandler);
   }
756 /**
     * Setup group table so that it reflects the current state of the group directory.
758  * Creation date: (17-07-00 17:00:48)
     */
760 public void initGroupTable()
   {
762   WorkGroup workGroup = new WorkGroup();
      tableModel = new javax.swing.table.DefaultTableModel();
764   ivjScrollPaneTable.setModel( tableModel );

766   String columnNames[] = WorkGroup.getFieldNames();
      for ( int i = 0 ; i < columnNames.length; i++ )
768   {
         tableModel.addColumn( columnNames[i] );
770   }
      Collection values = serverManager.getServerInfo().getGroups().values();
772   Iterator iterator = values.iterator();
      while ( iterator.hasNext() )
774   {
         workGroup = (WorkGroup)iterator.next();
776      Vector v = workGroup.getGroupAsVector();
         tableModel.addRow( v );
```

```
778     }

780  }
     /**
782   * Initialize the class.
      */
784  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
786     try {
           // user code begin {1}
788        // user code end
           setName("ServerApplication");
790        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setTitle("PeerView␣server");
792        setSize(598, 528);
           setJMenuBar(getServerApplicationJMenuBar());
794        setContentPane(getJFrameContentPane());
           initConnections();
796     } catch (java.lang.Throwable ivjExc) {
           handleException(ivjExc);
798     }
        // user code begin {2}
800     getServerManager().setServerApplication( this );
        getServerManager().configure();
802     center();
        initTable();
804     updateTable();
        // user code end
806  }
     /**
808   * Insert the method's description here.
      * Creation date: (04-07-00 23:11:34)
810   */
     public void initTable()
812  {
        WorkGroup workGroup = new WorkGroup();
814     tableModel = new javax.swing.table.DefaultTableModel();
        ivjScrollPaneTable.setModel( tableModel );
816
        String columnNames[] = WorkGroup.getFieldNames();
818     for ( int i = 0 ; i < columnNames.length; i++ )
        {
820        tableModel.addColumn( columnNames[i] );
        }
822  }
     /**
824   * main entrypoint - starts the part when it is run as an application
      * @param args java.lang.String[]
826   */
     public static void main(java.lang.String[] args)
828  {
        parseCommandLineArguments( args );
830     if ( ServerInfo.isVisible() == false )
        {
832        ServerManager serverManager = new ServerManager();
           serverManager.configure();
834        return;
        }
836     try {
           ServerApplication aServerApplication;
838        aServerApplication = new ServerApplication();
           aServerApplication.addWindowListener(new java.awt.event.WindowAdapter() {
840           public void windowClosing(java.awt.event.WindowEvent e) {
                 System.exit(0);
842           };
           });
```

```
844         aServerApplication.setVisible( ServerInfo.isVisible() );
        } catch (Throwable exception) {
846         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JFrame");
            exception.printStackTrace(System.out);
848     }
    }
850 /**
     * Parse the array of command line arguments in compliance with the POSIX specifications.
852  * Creation date: (10-09-00 07:54:32)
     * @param args java.lang.String[]
854  */
    public static void parseCommandLineArguments(String[] args)
856 {
        for ( int i = 0 ; i < args.length; i++ )
858     {
            if ( args[i].equalsIgnoreCase( ServerConstants.getINVISIBLE_ARGUMENT() ))
860         {
                ServerInfo.setVisible( false );
862             System.err.println( ServerConstants.getINVISIBLE_MODE_MESSAGE() );
            }
864         else if ( args[i].equalsIgnoreCase( ServerConstants.getSOCKET_ARGUMENT() ))
            {
866             ServerInfo.getPreferences().put( ServerConstants.getCONNECTION_TYPE(), ServerConstants.getSOCKET
                    () );
            }
868         else if ( args[i].equalsIgnoreCase( ServerConstants.getHTTP_ARGUMENT() ))
            {
870             ServerInfo.getPreferences().put( ServerConstants.getCONNECTION_TYPE(), ServerConstants.getHTTP()
                     );
            }
872     }
    }
874 /**
     * Insert the method's description here.
876  * Creation date: (29-06-00 18:47:13)
     * @param newServerManager serverapp.ServerManager
878  */
    public void setServerManager(ServerManager newServerManager) {
880     serverManager = newServerManager;
    }
882 /**
     * Insert the method's description here.
884  * Creation date: (17-07-00 21:34:13)
     * @param newTableModel javax.swing.table.DefaultTableModel
886  */
    public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
888     tableModel = newTableModel;
    }
890 /**
     * Insert the method's description here.
892  * Creation date: (17-08-00 00:22:47)
     */
894 public void terminateAndClose()
    {
896     getServerManager().terminate();
        dispose();
898 }
    /**
900  * Update the table underlying the server copy of the group directory.
     * Creation date: (22-07-00 23:35:40)
902  */
    public void updateTable()
904 {
        // clear the table
906     for ( int i = 0; i < tableModel.getRowCount(); i++ )
        {
```

```
      tableModel.removeRow( 0 );
    }
    tableModel.setNumRows( 0 );
    // update with contents of group directory
    Collection values = getServerManager().getServerInfo().getGroups().values();
    Iterator iterator = values.iterator();
    while ( iterator.hasNext() )
    {
      WorkGroup workGroup = (WorkGroup) iterator.next();
      Vector v = workGroup.getGroupAsVector();
      tableModel.addRow( v );
    }
    // Distribute surplus space evenly
    getScrollPaneTable().sizeColumnsToFit( -1 );
    getScrollPaneTable().revalidate();
    getScrollPaneTable().repaint();
    getJScrollPane1().revalidate();
    getJScrollPane1().repaint();
  }
  }
```

Listing C.30: ServerConstants.java

```
package serverapp;

 import javax.swing.border.*;
 import javax.swing.BorderFactory.*;
 import peerviewmisc.SharedConstants;
 import java.util.Properties;

 /**
 * This class holds constants specific to the PeerView server as well as those inherited from {@link #
     peerviewmisc.SharedConstants SharedConstants}.
 *
 * Creation date: (27-06-00 20:58:01)
 * @author:
 */

public class ServerConstants extends SharedConstants
{
   private final static java.lang.String PREFERENCES_FILE_NAME = "sprefs.ini";
   private final static java.lang.String GROUPS_FILE_NAME = "groups.dat";
   private final static java.lang.String GROUP_TABLE_CAST_FAILURE_MESSAGE = "Could␣not␣complete␣read␣of␣
       group␣table␣from␣disk";
   private final static java.lang.String ROOT_NODE_MESSAGE_TITLE = "System␣notice";
   private final static java.lang.String PROPERTIES_HEADER = "";
   private final static java.lang.String FRAME_HORIZONTAL_SIZE = "Frame␣horizontal␣size";
   private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame␣vertical␣size";
   private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";
   private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";
   private final static java.lang.String FRAME_X_COORDINATE = "Frame␣x␣coordinate";
   private final static java.lang.String FRAME_Y_COORDINATE = "Frame␣y␣coordinate";
   private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";
   private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";
   final static String DEFAULT_PREFERENCES[][] = { {SERVER_NAME, DEFAULT_SERVER_NAME},
                                    {SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT)},
                                    {CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE},
                                    {FRAME_HORIZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE},
                                    {FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE},
                                    {FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE},
                                    {FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE},
                                    {getREGISTRY_PORT(), getDEFAULT_REGISTRY_PORT()},
                                  };

   private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.5;
   private final static java.lang.String MESSAGES_FILE_NAME = "messages.dat";
```

```
42    private final static java.lang.String COULD_NOT_READ_MESSAGES = "Could␣not␣read␣the␣message␣table␣
          from␣disk.";
      private final static java.lang.String COULD_NOT_WRITE_MESSAGES = "Could␣not␣write␣the␣message␣table␣
          to␣disk.";
44    private final static java.lang.String COULD_NOT_READ_GROUPS = "Could␣not␣read␣the␣group␣directory␣
          from␣disk.";
      private final static java.lang.String COULD_NOT_WRITE_GROUPS = "Could␣not␣write␣the␣group␣directory␣
          to␣disk.";
46    private final static java.lang.String INVISIBLE_ARGUMENT = "-invisible";
      private final static java.lang.String INVISIBLE_MODE_MESSAGE = "Running␣in␣invisible␣mode␣-␣the␣GUI␣
          will␣be␣disabled.";
48    private final static java.lang.String COULD_NOT_CLOSE_GROUP_FILE = "Could␣not␣close␣group␣directory␣
          file.␣The␣directory␣may␣not␣have␣been␣properly␣initialized.";
    /**
50   * ServerConstants constructor comment.
     */
52  public ServerConstants() {
      super();
54  }
    /**
56   * Insert the method's description here.
     * Creation date: (05-10-00 09:48:05)
58   * @return java.lang.String
     */
60  public final static java.lang.String getCOULD_NOT_CLOSE_GROUP_FILE() {
      return COULD_NOT_CLOSE_GROUP_FILE;
62  }
    /**
64   * Insert the method's description here.
     * Creation date: (15-08-00 08:41:01)
66   * @return java.lang.String
     */
68  public final static java.lang.String getCOULD_NOT_READ_GROUPS() {
      return COULD_NOT_READ_GROUPS;
70  }
    /**
72   * Insert the method's description here.
     * Creation date: (14-08-00 23:24:44)
74   * @return java.lang.String
     */
76  public final static java.lang.String getCOULD_NOT_READ_MESSAGES() {
      return COULD_NOT_READ_MESSAGES;
78  }
    /**
80   * Insert the method's description here.
     * Creation date: (15-08-00 08:42:08)
82   * @return java.lang.String
     */
84  public final static java.lang.String getCOULD_NOT_WRITE_GROUPS() {
      return COULD_NOT_WRITE_GROUPS;
86  }
    /**
88   * Insert the method's description here.
     * Creation date: (14-08-00 23:29:55)
90   * @return java.lang.String
     */
92  public final static java.lang.String getCOULD_NOT_WRITE_MESSAGES() {
      return COULD_NOT_WRITE_MESSAGES;
94  }
    /**
96   * Insert the method's description here.
     * Creation date: (05-08-00 19:49:43)
98   * @return java.lang.String
     */
100 public final static java.lang.String getDEFAULT_FRAME_HORIZONTAL_SIZE() {
      return DEFAULT_FRAME_HORIZONTAL_SIZE;
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:50:26)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
   return DEFAULT_FRAME_VERTICAL_SIZE;
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:51:49)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
   return DEFAULT_FRAME_X_COORDINATE;
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:52:26)
 * @return java.lang.String
 */
public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
   return DEFAULT_FRAME_Y_COORDINATE;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 14:34:27)
 * @return java.util.Properties
 */
public static java.util.Properties getDEFAULT_PREFERENCES()
{
   Properties p = new Properties();

   for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
   {
      p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
   }
   return p;
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:49:06)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
   return FRAME_HORIZONTAL_SIZE;
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 20:05:17)
 * @return int
 */
public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
   return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
}
/**
 * Insert the method's description here.
 * Creation date: (05-08-00 19:49:24)
 * @return java.lang.String
 */
public final static java.lang.String getFRAME_VERTICAL_SIZE() {
   return FRAME_VERTICAL_SIZE;
}
/**
 * Insert the method's description here.
```

```
168   * Creation date: (05-08-00 19:51:18)
      * @return java.lang.String
170   */
      public final static java.lang.String getFRAME_X_COORDINATE() {
172     return FRAME_X_COORDINATE;
      }
174   /**
      * Insert the method's description here.
176   * Creation date: (05-08-00 19:51:34)
      * @return java.lang.String
178   */
      public final static java.lang.String getFRAME_Y_COORDINATE() {
180     return FRAME_Y_COORDINATE;
      }
182   /**
      * Insert the method's description here.
184   * Creation date: (04-07-00 15:57:55)
      * @return java.lang.String
186   */
      public final static java.lang.String getGROUP_TABLE_CAST_FAILURE_MESSAGE() {
188     return GROUP_TABLE_CAST_FAILURE_MESSAGE;
      }
190   /**
      * Insert the method's description here.
192   * Creation date: (04-07-00 15:19:40)
      * @return java.lang.String
194   */
      public final static java.lang.String getGROUPS_FILE_NAME() {
196     return GROUPS_FILE_NAME;
      }
198   /**
      * Insert the method's description here.
200   * Creation date: (10-09-00 07:52:35)
      * @return java.lang.String
202   */
      public final static java.lang.String getINVISIBLE_ARGUMENT() {
204     return INVISIBLE_ARGUMENT;
      }
206   /**
      * Insert the method's description here.
208   * Creation date: (10-09-00 08:08:58)
      * @return java.lang.String
210   */
      public final static java.lang.String getINVISIBLE_MODE_MESSAGE() {
212     return INVISIBLE_MODE_MESSAGE;
      }
214   /**
      * Insert the method's description here.
216   * Creation date: (14-08-00 23:19:47)
      * @return java.lang.String
218   */
      public final static java.lang.String getMESSAGES_FILE_NAME() {
220     return MESSAGES_FILE_NAME;
      }
222   /**
      * Insert the method's description here.
224   * Creation date: (03-07-00 14:43:22)
      * @return java.lang.String
226   */
      public final static java.lang.String getPREFERENCES_FILE_NAME() {
228     return PREFERENCES_FILE_NAME;
      }
230   /**
      * Insert the method's description here.
232   * Creation date: (05-08-00 19:44:31)
      * @return java.lang.String
```

```
 */
public final static java.lang.String getPROPERTIES_HEADER() {
   return PROPERTIES_HEADER;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:50:05)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */

public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
     Date creationDate)
{
   return "This␣is␣the␣discussion␣forum␣for␣the␣document␣" + documentName + "␣which␣was␣added␣by␣" +
        senderName +
         "␣on␣" + creationDate.toString();
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:50:05)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */

public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
     GregorianCalendar creationDate)
{
   return "This␣is␣the␣discussion␣forum␣for␣the␣document␣" + documentName + "␣which␣was␣added␣by␣" +
        senderName +
         "␣on␣" + creationDate.getTime().toString();
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:48:37)
 * @return java.lang.String
 */
public final static java.lang.String getROOT_NODE_MESSAGE_TITLE() {
   return ROOT_NODE_MESSAGE_TITLE;
}
}
```

Listing C.31: ServerInfo.java

```
package serverapp;

 import java.util.Properties;
 import java.util.Hashtable;
 import peerviewmisc.*;
 import java.io.*;

 /**
 * The PeerView server main entity object. This class holds data structures and fields shared by several
      server components.
 * Creation date: (28-06-00 22:16:34)
 * @author:
 */

public class ServerInfo
{
   private com.sun.media.jsdt.Channel groupManagementChannel = null;
   private java.util.Hashtable groups = new Hashtable();
   private static java.util.Properties preferences = null;
   private com.sun.media.jsdt.Session groupManagementSession;
   private int mostRecentPortNumber = ServerConstants.getDEFAULT_SERVER_PORT();
   private java.util.Hashtable discussions = new Hashtable();
   private java.util.Hashtable messages = new Hashtable();
```

```
      private static boolean visible = true;
24  /**
     * ServerInfo constructor comment.
26   */
    public ServerInfo()
28  {
      super();
30  }
    /**
32   * Insert the method's description here.
     * Creation date: (14-08-00 23:05:59)
34   * @exception java.lang.Throwable The exception description.
     */
36  protected void finalize() throws java.lang.Throwable
    {
38  }
    /**
40   * Insert the method's description here.
     * Creation date: (09-07-00 15:03:03)
42   * @return java.util.Hashtable
     */
44  public java.util.Hashtable getDiscussions() {
      return discussions;
46  }
    /**
48   * Insert the method's description here.
     * Creation date: (01-07-00 01:20:55)
50   * @return com.sun.media.jsdt.Channel
     */
52  public com.sun.media.jsdt.Channel getGroupManagementChannel() {
      return groupManagementChannel;
54  }
    /**
56   * Insert the method's description here.
     * Creation date: (04-07-00 17:12:12)
58   * @return com.sun.media.jsdt.Session
     */
60  public com.sun.media.jsdt.Session getGroupManagementSession() {
      return groupManagementSession;
62  }
    /**
64   * Insert the method's description here.
     * Creation date: (03-07-00 12:11:40)
66   * @return java.util.Properties
     */
68  public java.util.Hashtable getGroups()
    {
70    if ( groups == null )
      {
72      groups = new Hashtable();
      }
74    return groups;
    }
76  /**
     * Insert the method's description here.
78   * Creation date: (10-07-00 01:27:49)
     * @return java.util.Hashtable
80   */
    public java.util.Hashtable getMessages() {
82    return messages;
    }
84  /**
     * Insert the method's description here.
86   * Creation date: (07-07-00 17:30:38)
     * @return int
88   */
```

```
public int getMostRecentPortNumber() {
   return mostRecentPortNumber;
}
/**
 * Insert the method's description here.
 * Creation date: (02-07-00 22:08:51)
 * @return java.util.Properties
 */
public static java.util.Properties getPreferences()
{
   if ( preferences == null)
   {
      preferences = new Properties( ServerConstants.getDEFAULT_PREFERENCES() );
      readPreferences();
   }
   return preferences;
}
/**
 * Insert the method's description here.
 * Creation date: (10-09-00 07:59:29)
 * @return boolean
 */
public static boolean isVisible() {
   return visible;
}
/**
 * Read preferences from Properties object on disk. Each call to this function results in a read
 * from disk rather than an internal lookup, i.e. the values read from disk are not cached as a class
     member.
 * This eliminates data redundancy and ensures consistency. The overhead is neglible as the file in
     question is essentially
 * plain text.
 * Algorithm:
 * 1. Construct new Properties object P
 * 2. Initialize P with default preferences
 * 3. Read user preferences from disk file
 * 4. IF read is successful THEN
 * 4.1. Store user preferences in P
 * 5. Return P
 * This approach will return a valid object even if the read fails or in case of the application being
     used for the first
 * time.
 * Creation date: (25-06-00 13:54:55)
 */
public static void readPreferences()
{
   try
   {
      FileInputStream f = new FileInputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
      getPreferences().load(f);
      f.close();
   }
   catch (FileNotFoundException ffe)
   {
      // the exception can be handled by ignoring it since the default values in p will substitute for
          what should
      // have been read from file.
   }
   catch (IOException ie)
   {
      // any IOException != FileNotFound should be dealt with in the conventional manner
      System.err.println( ie.toString() );
      System.exit(0);
   }
}
/**
```

```
     * Insert the method's description here.
152  * Creation date: (09-07-00 15:03:03)
     * @param newDiscussions java.util.Hashtable
154  */
    public void setDiscussions(java.util.Hashtable newDiscussions) {
156    discussions = newDiscussions;
    }
158 /**
     * Insert the method's description here.
160  * Creation date: (01-07-00 01:20:55)
     * @param newGroupManagementChannel com.sun.media.jsdt.Channel
162  */
    public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
164    groupManagementChannel = newGroupManagementChannel;
    }
166 /**
     * Insert the method's description here.
168  * Creation date: (04-07-00 17:12:12)
     * @param newGroupManagementSession com.sun.media.jsdt.Session
170  */
    public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
172    groupManagementSession = newGroupManagementSession;
    }
174 /**
     * Insert the method's description here.
176  * Creation date: (03-07-00 12:11:40)
     * @param newGroups java.util.Properties
178  */
    public void setGroups(java.util.Hashtable newGroups) {
180    groups = newGroups;
    }
182 /**
     * Insert the method's description here.
184  * Creation date: (10-07-00 01:27:49)
     * @param newMessages java.util.Hashtable
186  */
    public void setMessages(java.util.Hashtable newMessages) {
188    messages = newMessages;
    }
190 /**
     * Insert the method's description here.
192  * Creation date: (07-07-00 17:30:38)
     * @param newMostRecentPortNumber int
194  */
    public void setMostRecentPortNumber(int newMostRecentPortNumber) {
196    mostRecentPortNumber = newMostRecentPortNumber;
    }
198 /**
     * Insert the method's description here.
200  * Creation date: (03-07-00 14:38:45)
     * @param newPreferences java.util.Properties
202  */
    public void setPreferences(java.util.Properties newPreferences) {
204    preferences = newPreferences;
    }
206 /**
     * Insert the method's description here.
208  * Creation date: (10-09-00 07:59:29)
     * @param newVisible boolean
210  */
    public static void setVisible(boolean newVisible) {
212    visible = newVisible;
    }
214 /**
     * Write preferences to disk file.
216  * Creation date: (17-07-00 12:13:32)
```

```
 * @param preferences java.util.Properties
 */
public static void writePreferences()
{
   // write Properties object to disk
   try
   {
      FileOutputStream f = new FileOutputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
      getPreferences().store( f, ServerConstants.getPROPERTIES_HEADER() );
      f.close();
   }
   catch (IOException ie)
   {
      System.err.println( ie.toString() );
      System.exit(0);
   }

}
}
```

Listing C.32: ServerManager.java

```
package serverapp;

 import com.sun.media.jsdt.*;
 import java.io.*;
 import java.util.Hashtable;
 import peerviewmisc.*;
 import java.util.Collection;
 import java.util.Iterator;
 import javax.swing.tree.*;
 import java.util.Date;
 import com.sun.media.jsdt.event.*;

/**
 * This is the main control class for the PeerView server application. It contains methods and control
     structures shared
 * by several server components.
 * Creation date: (27-06-00 20:17:35)
 * @author:
 */
public class ServerManager
{
   private ServerInfo serverInfo = null;

   public class DocNameAndSize implements Serializable
   {
      public String docName = null;
      public long size = 0;

      public DocNameAndSize( String docNameArg, long sizeArg )
      {
         docName = docNameArg;
         size = sizeArg;
      }
      public String getDocName()
      {
         return docName;
      }
      public void setDocName( String newDocName )
      {
         docName = newDocName;
      }
      public long getSize()
      {
         return size;
      }
```

```
      public void setSize( long newSize )
      {
         size = newSize;
      }
   };

   public class GroupListener extends SessionAdaptor
   {
      public String workGroupID = null;
      public GroupListener( String workGroupIDArg )
      {
         workGroupID = workGroupIDArg;
      }

      public void sessionJoined( SessionEvent se )
      {
         clientJoinedGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID
               ));
      }

      public void sessionLeft( SessionEvent se )
      {
         System.out.println( "Exit␣client:␣" + se.getClientName() );
         try
         {
         for ( int i = 0 ; i < se.getSession().listClientNames().length; i++ )
         {
            System.out.println( se.getSession().listClientNames()[i] );
         }
         }
         catch ( Exception E )
         {
         }
         clientLeftGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID ))
               ;
      }
   }

   public class DefaultClient implements com.sun.media.jsdt.Client
   {
      String name;

      public DefaultClient (String nameArg )
      {
         name = nameArg;
      }
      public Object authenticate ( com.sun.media.jsdt.AuthenticationInfo ai )
      {
         return null;
      }
      public String getName ()
      {
         return name;
      }
   };

   public class GroupManagementServer implements Client, ChannelConsumer
   {
      String name;

      public GroupManagementServer()
      {
      }

      public String getName()
      {
```

```
        return name;
110     }

112     public void setName( String newName )
        {
114        name = newName;
        }
116
        public Object authenticate ( AuthenticationInfo ai )
118     {
           return null;
120     }

122     public synchronized void dataReceived ( Data data )
        {
124        DataPackage incomingPackage = null;
           try
126        {
              incomingPackage = (DataPackage)data.getDataAsObject();
128        }
           catch ( Exception E )
130        {
              System.err.println( E.toString() );
132           System.exit(0);
           }
134        System.out.println( "Server␣fik␣" + incomingPackage.getProperties().getProperty( DataPackage.
                getDESCRIPTION() ));
           // Determine the type of incoming package
136        if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREQUEST_GROUP_DIRECTORY() ))
           {
138           sendGroupDirectory ( data.getSenderName() );
           }
140
           else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREQUEST_ADD_GROUP() ))
142        {
              addGroup( (peerviewmisc.WorkGroup) incomingPackage.getContents() );
144        }

146        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getCLIENT_ADDED_DOCUMENTS() ))
           {
148           createDiscussionForum( (String) incomingPackage.getProperties().getProperty( DataPackage.
                  getAUTHOR_NAME() ), (String) incomingPackage.getContents(), data.getSenderName() );
           }
150
           else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREQUEST_TOPIC_TREE() ))
152        {
              sendTopicTree( (String) incomingPackage.getContents(), data.getSenderName() );
154        }

156        else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREQUEST_MESSAGE() ))
           {
158           sendMessage( (String) incomingPackage.getProperties().getProperty( DataPackage.getMESSAGE_ID
                  () ), data.getSenderName() );
           }
160
           else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getCLIENT_ADDED_MESSAGE() ))
162        {
              addMessage( (Message) incomingPackage.getContents() );
164        }
```

```
166         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getTOPIC_TREE_UPDATE() ))
            {
168           updateTopicTree( (String) incomingPackage.getProperties().getProperty( DataPackage.getDOCPATH
                  () ), (DefaultTreeModel) incomingPackage.getContents() );
            }
170
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getDELETE_MESSAGE() ))
172         {
              deleteMessage( (String) incomingPackage.getContents() );
174         }
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getWORKGROUP_UPDATE_FROM_CLIENT() ))
176         {
              updateWorkGroup( (WorkGroup) incomingPackage.getContents() );
178         }
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREQUEST_REMOVE_GROUP() ))
180         {
              removeWorkGroup( (WorkGroup) incomingPackage.getContents() );
182         }
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getNEW_DOCUMENTS() ))
184         {
              clientAddedDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty(
                  DataPackage.getAUTHOR_NAME() ), incomingPackage.getProperties().getProperty( DataPackage
                  .getWORKGROUP_ID() ), (java.util.Vector) incomingPackage.getContents() );
186           broadcastGroupDirectory();
            }
188         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getREMOVE_DOCUMENTS() ))
            {
190           clientRemovedDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty(
                  DataPackage.getWORKGROUP_ID() ), (java.util.Vector) incomingPackage.getContents() );
              broadcastGroupDirectory();
192         }
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getCLIENT_LEFT_GROUP() ))
194         {
              clientLeftGroup( data.getSenderName(), (WorkGroup) getServerInfo().getGroups().get(
                  incomingPackage.getProperties().getProperty( DataPackage.getWORKGROUP_ID() )));
196         }
            else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase( ServerConstants.getCLIENT_LEFT() ))
198         {
              clientLeft( data.getSenderName() );
200         };

202       }
        };
204
        private PeerViewServer defaultServer = new PeerViewServer(ServerConstants.getDEFAULT_SERVER_NAME());
206     private GroupManagementServer groupManagementServer = null;
        private ServerApplication serverApplication = null;
208   /**
       * ServerManager constructor comment.
210    */
      public ServerManager() {
212     super();
        initialize();
214   }
      /**
216    * .
       * Creation date: (07-07-00 02:16:02)
218    * @param group peerviewmisc.WorkGroup
```

```
 */
public void activateGroup(WorkGroup workGroup)
{
     Session session = null;
     URLString url = null;
     GroupManagementServer newServer = new GroupManagementServer();
     newServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME() ));

     try
     {
     boolean created = false;
/*   // Cycle through port numbers until an unoccupied one is found and use it for creating a new
    session
     while ( created == false )
        {
        try
        {
*/         url = URLString.createSessionURL( ServerInfo.getPreferences().getProperty ( ServerConstants.
    getSERVER_NAME() ),
                                      Integer.parseInt( ServerInfo.getPreferences().getProperty(
                                          ServerConstants.getSERVER_PORT() )),
                                      ServerInfo.getPreferences().getProperty ( ServerConstants.
                                          getCONNECTION_TYPE() ),
                                      workGroup.getWorkGroupID() );
           // System.out.println( workGroup.getWorkGroupID() );
           session = SessionFactory.createSession( newServer, url, true );
           created = true;
           session.addSessionListener( new GroupListener( workGroup.getWorkGroupID() ));
/*      }
        catch ( PortInUseException PIUE )
        {
           serverInfo.setMostRecentPortNumber( serverInfo.getMostRecentPortNumber() + 1 );
        }

     }
*/   Channel groupChannel = session.createChannel( newServer, workGroup.getWorkGroupChannelID(), true,
    true, true );
     groupChannel.addConsumer( newServer, newServer );
     workGroup.setSessionURL( url );
     }
     catch ( JSDTException JSDTE )
     {
        handleJSDTException( JSDTE );
     }

}
/**
 * Activate the groups read from disk file by creating and registering the appropriate sessions and
     channels.
 * Creation date: (05-07-00 23:33:46)
 */
public void activateGroups()
{
   Collection values = serverInfo.getGroups().values();

   Iterator iterator = values.iterator();
   while ( iterator.hasNext() )
   {
      activateGroup( (WorkGroup) iterator.next() );

   }
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 19:05:53)
 * @param workGroup peerviewmisc.WorkGroup
```

```
 */
public void addGroup(peerviewmisc.WorkGroup workGroup)
{
   serverInfo.getGroups().put( workGroup.getWorkGroupID(), workGroup);
   activateGroup( workGroup );
   broadcastGroupDirectory();
   serverApplication.updateTable();
}
/**
 * Insert the method's description here.
 * Creation date: (11-07-00 21:12:23)
 * @param peerviewmisc.Message
 */
public void addMessage(Message newMessage)
{
   getServerInfo().getMessages().put( newMessage.getMessageID(), newMessage );
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 03:03:54)
 * @param senderName java.lang.String
 * @param documents java.util.Vector
 */
public void broadcastDocumentRemovals(String senderName, WorkGroup workGroup, java.util.Vector documents
      )
{
   DataPackage outgoingPackage = new DataPackage();
   outgoingPackage.setContents( documents );
   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
        getREMOVE_DOCUMENTS() );
   outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), workGroup.getWorkGroupID
        () );
   outgoingPackage.getProperties().setProperty( DataPackage.getCLIENT_NAME(), senderName );
   outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ServerInfo.getPreferences
        ().getProperty( ServerConstants.getSERVER_NAME() ));
   try
   {
      URLString url = workGroup.getSessionURL();
      Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
      Channel groupChannel = session.createChannel( getGroupManagementServer(), workGroup.
           getWorkGroupChannelID(), true, true, false);
      Data data = new Data( outgoingPackage );
      groupChannel.sendToOthers( getGroupManagementServer(), data );
   }
   catch ( JSDTException JSDTE )
   {
      handleJSDTException( JSDTE );
   }
}
/**
 * Insert the method's description here.
 * Creation date: (22-07-00 20:08:48)
 * @param workGroup peerviewmisc.WorkGroup
 */
public void broadcastGroupDirectory()
{
   DataPackage outgoingPackage = new DataPackage();

   outgoingPackage.setContents( getServerInfo().getGroups() );
   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
        getGROUP_DIRECTORY() );
   try
   {
      if ( getServerInfo().getGroupManagementChannel() != null )
      {
         getServerInfo().getGroupManagementChannel().sendToOthers( getGroupManagementServer(),
```

```
                                                    new Data( outgoingPackage ));
340         }
      }
342   catch ( JSDTException JSDTE )
      {
344      handleJSDTException( JSDTE );
      }
346   if ( getServerApplication() != null )
      {
348      getServerApplication().updateTable();
      }
350 }
    /**
352  * Insert the method's description here.
     * Creation date: (27-08-00 20:58:01)
354  */
    public void broadcastTopicTree( String documentName, DefaultTreeModel treeUpdate)
356 {
      DataPackage outgoingPackage = new DataPackage();
358
      outgoingPackage.setContents( treeUpdate );
360   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
          getTOPIC_TREE_UPDATE() );
      outgoingPackage.getProperties().setProperty( DataPackage.getDOCPATH(), documentName );
362   try
      {
364      getServerInfo().getGroupManagementChannel().sendToOthers( getGroupManagementServer(),
                                                  new Data( outgoingPackage ));
366   }
      catch ( JSDTException JSDTE )
368   {
         handleJSDTException( JSDTE );
370   }

372
    }
374 /**
     * Insert the method's description here.
376  * Creation date: (19-09-00 18:40:28)
     * @param senderName java.lang.String
378  * @param groupID java.lang.String
     * @param documents java.util.Vector
380  */
    public void clientAddedDocuments(String senderName, String authorName, String groupID, java.util.Vector
        documents)
382 {
      WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupID );
384
      for ( int i = 0 ; i < documents.size(); i++ )
386   {
         CompressedDocumentPackage docPackage = (CompressedDocumentPackage) documents.get( i );
388      Pair docNameAndSize = new Pair( docPackage.getPath(), String.valueOf( docPackage.
             getUncompressedLength() ));
         ( (java.util.Vector) groupToBeUpdated.getDocuments().get( senderName )).add( docNameAndSize );
390      groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + docPackage.
             getUncompressedLength() );
         groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() + 1 );
392   }
      getServerInfo().getGroups().put( groupID, groupToBeUpdated );
394 }
    /**
396  * Update the data structure holding the groups and submit it to all clients connected to the group
         management channel.
     * Creation date: (19-09-00 17:55:14)
398  * @param clientName java.lang.String
     * @param workGroup peerviewmisc.WorkGroup
```

```
 */
public void clientJoinedGroup(String clientName, WorkGroup workGroup)
{
   WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
         );

   groupToBeUpdated.getParticipants().add( clientName );
   groupToBeUpdated.getDocuments().put( clientName, new java.util.Vector() );
   getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
   broadcastGroupDirectory();
}
/**
 * Forcibly expel client from group management session if it has not already left. This to ensure
     consistency when clients
 * terminate abruptly, i.e. do not exit properly.
 * Creation date: (01-11-00 23:45:19)
 * @param clientName java.lang.String
 */
public void clientLeft(String clientName)
{
   try
   {
      for ( int i = 0 ; i < getServerInfo().getGroupManagementSession().listClientNames().length; i++ )
      {
         if ( getServerInfo().getGroupManagementSession().listClientNames()[i].equalsIgnoreCase(
              clientName ))
         {
               //System.out.println( "Found client : " + clientName );
               DefaultClient tempClient = new DefaultClient( clientName );
               getServerInfo().getGroupManagementSession().leave( tempClient );
               break;
         }
      }
   }
   catch ( JSDTException JSDTE )
   {
      handleJSDTException( JSDTE );
   }
}
/**
 * This method is triggered when a client leaves a group. The server updates the group's entry in the
     group directory
 * and resubmits it to other clients. Client that partake in the group that the client left are notified
      of which
 * documents to remove from their panoramas.
 * Creation date: (19-09-00 17:55:46)
 * @param clientName java.lang.String
 * @param workGroup peerviewmisc.WorkGroup
 */
public void clientLeftGroup(String clientName, WorkGroup workGroup)
{
   WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
         );

   if ( groupToBeUpdated.getParticipants().contains( clientName ))
   {
      java.util.Vector documentNames = new java.util.Vector();
      java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( clientName );
      Iterator docIterator = documents.iterator();
      while ( docIterator.hasNext() )
      {
         Pair docNameAndSize = (Pair) docIterator.next();
         groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt(
              docNameAndSize.getValue() ));
         groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
         documentNames.add( docNameAndSize.getKey() );
```

```
        }
460     broadcastDocumentRemovals( clientName, groupToBeUpdated, documentNames );

462     groupToBeUpdated.getParticipants().remove( clientName );
        groupToBeUpdated.getDocuments().remove( clientName );
464     getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
        broadcastGroupDirectory();
466   }
    }
468 /**
     * This method is triggered when a client removes one or more documents from his or her panorama. The
         server updates the
470  * group that the client belongs to and resubmits the group directory to all clients.
     * Creation date: (19-09-00 19:08:00)
472  * @param files java.util.HashSet
     */
474 public void clientRemovedDocuments(String senderName, String groupID, java.util.Vector filePaths)
    {
476   WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupID );

478   Iterator iterator = filePaths.iterator();
      while ( iterator.hasNext() )
480   {
        String path = (String) iterator.next();
482     java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( senderName );
        Iterator docIterator = documents.iterator();
484     Pair docNameAndSize = null;
        while ( docIterator.hasNext() )
486     {
          docNameAndSize = (Pair) docIterator.next();
488       if ( docNameAndSize.getKey().equals( path ))
          {
490         groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt
                ( docNameAndSize.getValue() ));
            groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
492         documents.remove( docNameAndSize );
            break;
494       }
        }
496   }
      getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
498 }
    /**
500  * Perform miscellaneous initialization operations.
     * Creation date: (10-10-00 23:12:57)
502  */
    public void configure()
504 {
      startRegistry();
506   createGroupManagement();
      readMessages();
508   readGroups();
      readDiscussions();
510 }
    /**
512  * Insert the method's description here.
     * Creation date: (29-06-00 08:57:23)
514  * @return java.lang.String
     * @param sessionName java.lang.String
516  */
    public String createClientName(String sessionName)
518 {
      return ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME() )+ sessionName;
520 }
    /**
```

```
522   * Create a new discussion forum and initialize it to the default structure, namely a root node and
          system notification
      * message.
524   * Creation date: (09-07-00 14:55:18)
      * @param newDocument java.io.File
526   */
      public void createDiscussionForum(String authorName, String newDocument, String senderName)
528   {
        String fqDocumentName = ServerConstants.createFullyQualifiedDocumentName( senderName, newDocument );
530     // if a discussion forum for this document does not already exist then create one
        if ( getServerInfo().getDiscussions().containsKey( fqDocumentName )== false )
532     {
          Message message = new Message();
534       message.setAuthor( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME() ));
          message.setCreationDate( new java.util.Date() );
536       message.setCreator( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME() ));
          message.setTitle( ServerConstants.getROOT_NODE_MESSAGE_TITLE() );
538       message.setContents( ServerConstants.getROOT_NODE_MESSAGE_CONTENTS( newDocument, authorName,
              message.getCreationDate() ));
          message.setDocument( newDocument );
540       getServerInfo().getMessages().put( message.getMessageID(), message );
                                                                          // udskift med konstant
542       DefaultMutableTreeNode innerNode = new DefaultMutableTreeNode( newDocument + ",␣authored␣by␣" +
              authorName, true );
          DefaultMutableTreeNode treeNode = new DefaultMutableTreeNode( message.getMessageID(), false );
544
          innerNode.add( treeNode );
546       getServerInfo().getDiscussions().put( fqDocumentName, new DefaultTreeModel( innerNode ));
          // System.out.println( "New topic tree: " + fqDocumentName );
548     }
      }
550   /**
      * Setup the group management session and channel and activate it.
552   * Creation date: (03-07-00 11:29:03)
      */
554   public void createGroupManagement()
      {
556     Session session = null;
        Channel groupRequestChannel = null;
558     try
        {
560
          URLString url = URLString.createSessionURL( (String)ServerInfo.getPreferences().getProperty(
              ServerConstants.getSERVER_NAME() ),
562                                   Integer.parseInt( ServerInfo.getPreferences().getProperty(
                                          ServerConstants.getSERVER_PORT() )),
                                      (String)ServerInfo.getPreferences().getProperty( ServerConstants.
                                          getCONNECTION_TYPE() ),
564                                   (String)ServerConstants.getGROUP_MANAGEMENT_SESSION_NAME() );
          session = SessionFactory.createSession( getGroupManagementServer(), url, true);
566       groupRequestChannel = session.createChannel( getGroupManagementServer(),
                                       ServerConstants.getGROUP_CHANNEL_NAME(),
568                                    true, true, true);
          groupRequestChannel.addConsumer( getGroupManagementServer(), getGroupManagementServer() );
570       DataPackage pack = new DataPackage();
          pack.setContents( null );
572       groupRequestChannel.sendToAll( getGroupManagementServer(), new Data( pack ));
          // System.out.println( url.toString() );
574     }
        catch (JSDTException JSDTE)
576     {
          handleJSDTException( JSDTE );
578     }
        getServerInfo().setGroupManagementSession( session );
580     getServerInfo().setGroupManagementChannel( groupRequestChannel );
      }
```

```
582  /**
      * Shut down a group.
584   * Creation date: (27-07-00 00:35:12)
      * @param workGroup peerviewmisc.WorkGroup
586   */
     public void deactivateGroup(WorkGroup workGroup)
588  {
          try
590       {
             URLString url = workGroup.getSessionURL();
592          Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
             Channel channel = session.createChannel( getGroupManagementServer(), workGroup.
                  getWorkGroupChannelID(), true, true, false );
594          channel.destroy( getGroupManagementServer() );
             session.destroy( getGroupManagementServer() );
596       }
          catch ( JSDTException JSDTE )
598       {
             handleJSDTException( JSDTE );
600       }
     }
602  /**
      * Insert the method's description here.
604   * Creation date: (20-10-00 15:06:46)
      */
606  public void deactivateGroups()
     {
608     Collection values = serverInfo.getGroups().values();

610     Iterator iterator = values.iterator();
        while ( iterator.hasNext() )
612     {
           deactivateGroup( (WorkGroup) iterator.next() );
614
        }
616  }
     /**
618   * Insert the method's description here.
      * Creation date: (12-07-00 12:14:10)
620   * @param messageID java.lang.String
      */
622  public void deleteMessage(String messageID)
     {
624     serverInfo.getMessages().remove( messageID );
     }
626  /**
      * Insert the method's description here.
628   * Creation date: (20-10-00 15:15:34)
      */
630  public void destroyGroupManagement()
     {
632     try
        {
634        getServerInfo().getGroupManagementChannel().destroy( getGroupManagementServer() );
           getServerInfo().getGroupManagementSession().destroy( getGroupManagementServer() );
636     }
        catch ( JSDTException JSDTE )
638     {
           handleJSDTException( JSDTE );
640     }
     }
642  /**
      * Insert the method's description here.
644   * Creation date: (04-07-00 16:11:19)
      */
646  protected void finalize()
```

```
{
}
/**
 * Insert the method's description here.
 * Creation date: (30-06-00 11:17:21)
 * @return serverapp.PeerViewServer
 */
public PeerViewServer getDefaultServer() {
   return defaultServer;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:23:42)
 * @return serverapp.GroupManagementServer
 */
public GroupManagementServer getGroupManagementServer()
{
   if (groupManagementServer == null)
   {
      groupManagementServer = new GroupManagementServer();
      groupManagementServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.
           getSERVER_NAME() ));
   }
   return groupManagementServer;
}
/**
 * Insert the method's description here.
 * Creation date: (02-09-00 08:46:31)
 * @return serverapp.ServerApplication
 */
public ServerApplication getServerApplication() {
   return serverApplication;
}
/**
 * Insert the method's description here.
 * Creation date: (28-06-00 22:59:39)
 * @return serverapp.ServerInfo
 */
public ServerInfo getServerInfo()
{
   if ( serverInfo == null )
   {
      serverInfo = new ServerInfo();
   }
   return serverInfo;
}
/**
 * Handle the different subclasses of JSDTException, i.e. the different types of exception that JSDT
     actions may throw,
 * by issuing appropriate messages and taking corrective or conclusive action.
 * Creation date: (08-08-00 16:34:28)
 * @param exception com.sun.media.jsdt.JSDTException
 */
public void handleJSDTException(JSDTException exception)
{
   String message = (String) ServerConstants.getJSDT_EXCEPTION_MESSAGES_TABLE().get( exception.getClass
        ().toString() );
   message += "␣(␣" + exception.toString() + "␣)";
   System.err.println( message );
}
/**
 * Insert the method's description here.
 * Creation date: (01-07-00 01:14:14)
 */
public void initialize()
{
```

```
710  }
     /**
712   * Read the discussion "database" (a text file, essentially) from disk and initialize the corresponding
          data structure.
      * Creation date: (24-08-00 08:45:43)
714   */
     public void readDiscussions()
716  {
        try
718     {
           FileInputStream fistream = new FileInputStream( ServerConstants.getDISCUSSIONS_FILE_NAME() );
720        ObjectInputStream oistream = new ObjectInputStream( fistream );
           Hashtable discussionsTable = (Hashtable) oistream.readObject();
722        if ( discussionsTable != null )
           {
724           getServerInfo().setDiscussions( discussionsTable );
           }
726        else
           {
728           getServerInfo().setDiscussions( new Hashtable() );
           }
730        // System.out.println( "Discussionstablesize: " + discussionsTable.size() );
           oistream.close();
732     }
        catch ( Exception e )
734     {
           System.err.println( ServerConstants.getCOULD_NOT_READ_MESSAGES() );
736     }

738  }
     /**
740   * Read the groups directory from persistent storage, i.e. disk file
      * Creation date: (04-07-00 15:03:54)
742   */
     public void readGroups()
744  {
        try
746     {
        FileInputStream fis = null;
748     ObjectInputStream ois = null;
        fis = new FileInputStream( ServerConstants.getGROUPS_FILE_NAME() );
750     ois = new ObjectInputStream( fis );

752  // while ( ois.available() > 0 )
     // {
754        // the below simulates a loop by trying a readObject operation until an exception is thrown. Not
              kosher, but
           // the only way that seems to work at the moment.
756        while (true)
           {
758           try
              {
760              WorkGroup workGroup = (WorkGroup) ois.readObject();
                 workGroup.setNumberOfDocuments( 0 );
762              workGroup.setTotalDataVolume( 0 );
                 workGroup.setDocuments( new java.util.Hashtable() );
764              workGroup.setParticipants( new java.util.HashSet() );
                 // System.out.println( "WorkGroup: " + workGroup.getWorkGroupID() );
766              getServerInfo().getGroups().put( workGroup.getWorkGroupID(), workGroup);
                 activateGroup( workGroup );
768           }
              catch ( Exception E )
770           {
                 break;
772           }
           }
```

```
774
    // }
776    ois.close();
       broadcastGroupDirectory();
778    }
       catch ( IOException E )
780    {
          System.err.println( ServerConstants.getCOULD_NOT_READ_GROUPS() );
782    }
    }
784 /**
     * Insert the method's description here.
786  * Creation date: (14-08-00 23:03:40)
     */
788 public void readMessages()
    {
790    try
       {
792       FileInputStream fistream = new FileInputStream( ServerConstants.getMESSAGES_FILE_NAME() );
          ObjectInputStream oistream = new ObjectInputStream( fistream );
794       Hashtable messageTable = (Hashtable) oistream.readObject();
          if ( messageTable != null )
796       {
             getServerInfo().setMessages( messageTable );
798       }
          else
800       {
             getServerInfo().setMessages( new Hashtable() );
802       }
          // System.out.println( "Messagetablesize: " + messageTable.size() );
804       oistream.close();
       }
806    catch ( Exception e )
       {
808       System.err.println( ServerConstants.getCOULD_NOT_READ_MESSAGES() );
       }
810
    }
812 /**
     * Insert the method's description here.
814  * Creation date: (27-07-00 00:31:10)
     * @param workGroup peerviewmisc.WorkGroup
816  */
    public void removeWorkGroup(WorkGroup workGroup)
818 {
       System.out.println( "I␣removeworkgroup" );
820    deactivateGroup( workGroup );
       System.out.println( "After␣deactivateGroup" );
822    getServerInfo().getGroups().remove( workGroup.getWorkGroupID() );
       System.out.println( "about␣to␣broadcast␣group␣directory␣from␣removeworkgroup" );
824    broadcastGroupDirectory();
    }
826 /**
     * Insert the method's description here.
828  * Creation date: (07-07-00 14:53:16)
     * @param recipientName java.lang.String
830  */
    public void sendGroupDirectory(String recipientName)
832 {

834    DataPackage outgoingPackage = new DataPackage();
       outgoingPackage.setContents( serverInfo.getGroups() );
836    outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
            getGROUP_DIRECTORY() );

838    try
```

```
      {
840   serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer,
                                        recipientName,
842                                     new Data( outgoingPackage ));
      System.out.println( "Sent␣group␣directory" );
844   }
      catch ( JSDTException JSDTE )
846   {
         handleJSDTException( JSDTE );
848   }

850
    }
852 /**
     * Insert the method's description here.
854  * Creation date: (10-07-00 02:20:49)
     * @param messageID java.lang.String
856  * @param recipientName java.lang.String
     */
858 public void sendMessage(String messageID, String recipientName)
    {
860   DataPackage outgoingPackage = new DataPackage();

862   // System.out.println( "Sent message: " + messageID );
      outgoingPackage.setContents( serverInfo.getMessages().get( messageID ));
864   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.getMESSAGE
          () );
      try
866   {
         serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer, recipientName, new
             Data( outgoingPackage ));
868   }
      catch ( JSDTException JSDTE )
870   {
         handleJSDTException( JSDTE );
872   }
    }
874 /**
     * Precondition: documentName is the fully qualified name of a document as specified in method
         SharedConstants.createFullyQualifiedDocumentName
876  * Creation date: (09-07-00 22:20:13)
     * @param documentName java.lang.String
878  * @param recipientName java.lang.String
     */
880 public void sendTopicTree(String documentName, String recipientName)
    {
882   DataPackage outgoingPackage = new DataPackage();

884   // System.out.println( "Sending topic tree: " + documentName );
      outgoingPackage.setContents( getServerInfo().getDiscussions().get( documentName ));
886   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
          getTOPIC_TREE() );
      outgoingPackage.getProperties().setProperty( DataPackage.getDOCPATH(), documentName );
888   try
      {
890      serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer, recipientName, new
             Data( outgoingPackage ));
      }
892   catch ( JSDTException JSDTE )
      {
894      handleJSDTException( JSDTE );
      }
896 }
    /**
898  * Insert the method's description here.
     * Creation date: (30-06-00 11:17:21)
```

```
900   * @param newDefaultServer serverapp.PeerViewServer
      */
902   public void setDefaultServer(PeerViewServer newDefaultServer) {
         defaultServer = newDefaultServer;
904   }
      /**
906    * Insert the method's description here.
       * Creation date: (03-07-00 12:23:42)
908    * @param newGroupManagementServer serverapp.GroupManagementServer
       */
910   public void setGroupManagementServer(GroupManagementServer newGroupManagementServer) {
         groupManagementServer = newGroupManagementServer;
912   }
      /**
914    * Insert the method's description here.
       * Creation date: (02-09-00 08:46:31)
916    * @param newServerApplication serverapp.ServerApplication
       */
918   public void setServerApplication(ServerApplication newServerApplication) {
         serverApplication = newServerApplication;
920   }
      /**
922    * Insert the method's description here.
       * Creation date: (28-06-00 22:59:39)
924    * @param newServerInfo serverapp.ServerInfo
       */
926   public void setServerInfo(ServerInfo newServerInfo) {
         serverInfo = newServerInfo;
928   }
      /**
930    * Insert the method's description here.
       * Creation date: (27-06-00 20:18:18)
932    */
      public void startRegistry()
934   {
         try
936      {
            RegistryFactory.registryPort = Integer.parseInt( ServerInfo.getPreferences().getProperty(
                 ServerConstants.getREGISTRY_PORT() ));
938         com.sun.media.jsdt.RegistryFactory.startRegistry(
                  (String) ServerInfo.getPreferences().getProperty( ServerConstants.getCONNECTION_TYPE() ));
940      //      Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getREGISTRY_PORT()
               )));
            if ( RegistryFactory.registryExists( ServerInfo.getPreferences().getProperty( ServerConstants.
                 getCONNECTION_TYPE() ),
942                                  Integer.parseInt( ServerInfo.getPreferences().getProperty(
                                          ServerConstants.getREGISTRY_PORT() ))));
            {
944             System.out.println( "Successfully␣started␣registry" );
            }
946
         }
948      catch (com.sun.media.jsdt.NoRegistryException nre)
         {
950         // System.out.println( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
                 getCONNECTION_TYPE() ));
            // System.out.println( ServerInfo.getPreferences().getProperty( ServerConstants.getREGISTRY_PORT()
                  ));
952         System.err.println( nre.toString() );
            System.exit(0);
954      }
         catch (com.sun.media.jsdt.RegistryExistsException ree)
956      {
            System.err.println( ree.toString() );
958         System.exit(0);
         }
```

```
960
    }
962 /**
     * Insert the method's description here.
964  * Creation date: (21-10-00 23:27:34)
     */
966 public void stopRegistry()
    {
968    try
       {
970    RegistryFactory.stopRegistry( ServerInfo.getPreferences().getProperty( ServerConstants.
           getCONNECTION_TYPE() ),
                            Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.
                                getSERVER_PORT() )));
972    }
       catch ( JSDTException JSDTE )
974    {
          handleJSDTException( JSDTE );
976    }
    }
978 /**
     * Insert the method's description here.
980  * Creation date: (21-10-00 23:20:36)
     */
982 public void terminate()
    {
984    deactivateGroups();
       writeDiscussions();
986    writeMessages();
       writeGroups();
988    getServerInfo().writePreferences();
       destroyGroupManagement();
990    try
       {
992       System.runFinalization();
          Runtime.getRuntime().exit( 0 );
994    }
       catch ( Exception E )
996    {
          System.out.println( ServerConstants.getCOULD_NOT_EXIT_PROPERLY() );
998    }
    }
1000 /**
      * Insert the method's description here.
1002  * Creation date: (11-07-00 21:37:44)
      * @param documentName java.lang.String
1004  * @param newTree javax.swing.tree.DefaultTreeModel
      */
1006 public void updateTopicTree(String documentName, DefaultTreeModel newTree)
     {
1008    getServerInfo().getDiscussions().put( documentName, newTree );
        broadcastTopicTree( documentName, newTree );
1010 }
     /**
1012  * Insert the method's description here.
      * Creation date: (22-07-00 18:40:42)
1014  * @param workGroup peerviewmisc.WorkGroup
      */
1016 public void updateWorkGroup(WorkGroup workGroup)
     {
1018    WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
            );

1020    groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() + workGroup.
            getNumberOfDocuments() );
```

```
      groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + workGroup.
          getTotalDataVolume() );
      getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
      broadcastGroupDirectory();
   }
   /**
    * Insert the method's description here.
    * Creation date: (24-08-00 08:39:39)
    */
   public void writeDiscussions()
   {
      try
      {
         FileOutputStream fostream = new FileOutputStream( ServerConstants.getDISCUSSIONS_FILE_NAME() );
         ObjectOutputStream oostream = new ObjectOutputStream( fostream );
         oostream.writeObject( getServerInfo().getDiscussions() );
         // System.out.println( "WroteDiscussions" );
         oostream.flush();
         oostream.close();
      }
      catch ( Exception e )
      {
         System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
      }

   }
   /**
    * Write registered groups to persistent storage, i.e. disk file
    * Creation date: (04-07-00 15:03:41)
    */
   public void writeGroups()
   {
      try
      {
         FileOutputStream fos = new java.io.FileOutputStream( ServerConstants.getGROUPS_FILE_NAME() );
         ObjectOutputStream oos = new ObjectOutputStream( fos );

         Collection values = serverInfo.getGroups().values();
         Iterator iterator = values.iterator();

         while ( iterator.hasNext() )
         {
            WorkGroup workGroup = (WorkGroup) iterator.next();
            workGroup.setParticipants( null );
            workGroup.setDocuments( null );
            oos.writeObject( workGroup );
      /*    oos.writeUTF( workGroup.getGroupName() );
            oos.writeInt( workGroup.getMaximumParticipants() );
            oos.writeUTF( workGroup.getDescription() );
            oos.writeUTF( workGroup.getCreationDate().toString() );
            oos.writeUTF( workGroup.getCreator() );
   */
            // System.out.println( "Wrote group" );
         }
         oos.flush();
         oos.close();
      }
      catch ( IOException IOE )
      {
         System.err.println( ServerConstants.getCOULD_NOT_WRITE_GROUPS() );

      }
   }
   /**
    * Insert the method's description here.
    * Creation date: (14-08-00 23:26:34)
```

```
 */
public void writeMessages()
{
   try
   {
      FileOutputStream fostream = new FileOutputStream( ServerConstants.getMESSAGES_FILE_NAME() );
      ObjectOutputStream oostream = new ObjectOutputStream( fostream );
      oostream.writeObject( getServerInfo().getMessages() );
      // System.out.println( "WroteMessages" );
      oostream.flush();
      oostream.close();
   }
   catch ( Exception e )
   {
      System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
   }
}
}
```

Listing C.33: ServerSetup.java

```
package serverapp;

import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.awt.*;
import java.util.Properties;
import java.io.*;
import java.awt.event.ActionListener;
import java.awt.event.*;
/*
 * The server setup dialog box.
 * Creation date: (28-05-00 20:58:57)
 * @author:
 */
public class ServerSetup extends JDialog
{
   public class OKButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         submitAndClose();
      }
   };

   public class CancelButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         cancelAndClose();
      }
   };


   private JButton ivjCancelButton = null;
   private JPanel ivjJDialogContentPane = null;
   private JPanel ivjJPanel1 = null;
   private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
   private JButton ivjOKButton = null;
   private JComboBox ivjProtocolComboBox = null;
   private JLabel ivjProtocolLabel = null;
   private JTextField ivjServerNameField = null;
   private JLabel ivjServerNameLabel = null;
   private JLabel ivjServerPortLabel = null;
   private JTextField ivjServerPortField = null;
   private JPanel ivjTopPanel = null;
```

```
/**
 * ServerSetup constructor comment.
 */
public ServerSetup() {
   super();
   initialize();
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 */
public ServerSetup(java.awt.Dialog owner) {
   super(owner);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public ServerSetup(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public ServerSetup(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public ServerSetup(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 */
public ServerSetup(java.awt.Frame owner) {
   super(owner);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public ServerSetup(java.awt.Frame owner, String title) {
   super(owner, title);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public ServerSetup(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * ServerSetup constructor comment.
 * @param owner java.awt.Frame
```

```
     * @param modal boolean
114   */
     public ServerSetup(java.awt.Frame owner, boolean modal) {
116     super(owner, modal);
     }
118  /**
     * Insert the method's description here.
120  * Creation date: (05-08-00 20:54:45)
     */
122  public void cancelAndClose()
     {
124     dispose();
     }
126  /**
     * Return the CancelButton property value.
128  * @return javax.swing.JButton
     */
130  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JButton getCancelButton() {
132     if (ivjCancelButton == null) {
           try {
134           ivjCancelButton = new javax.swing.JButton();
              ivjCancelButton.setName("CancelButton");
136           ivjCancelButton.setText("Cancel");
              // user code begin {1}
138           ivjCancelButton.addActionListener( new CancelButtonHandler() );
              // user code end
140        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
142           // user code end
              handleException(ivjExc);
144        }
        }
146     return ivjCancelButton;
     }
148  /**
     * Return the JDialogContentPane property value.
150  * @return javax.swing.JPanel
     */
152  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJDialogContentPane() {
154     if (ivjJDialogContentPane == null) {
           try {
156           ivjJDialogContentPane = new javax.swing.JPanel();
              ivjJDialogContentPane.setName("JDialogContentPane");
158           ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
              getJDialogContentPane().add(getJPanel1(), "South");
160           getJDialogContentPane().add(getTopPanel(), "Center");
              // user code begin {1}
162           // user code end
           } catch (java.lang.Throwable ivjExc) {
164           // user code begin {2}
              // user code end
166           handleException(ivjExc);
           }
168     }
        return ivjJDialogContentPane;
170  }
     /**
172  * Return the JPanel1 property value.
     * @return javax.swing.JPanel
174  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
176  private javax.swing.JPanel getJPanel1() {
        if (ivjJPanel1 == null) {
178        try {
```

```
            ivjJPanel1 = new javax.swing.JPanel();
180         ivjJPanel1.setName("JPanel1");
            ivjJPanel1.setLayout(getJPanel1FlowLayout());
182         getJPanel1().add(getOKButton(), getOKButton().getName());
            getJPanel1().add(getCancelButton(), getCancelButton().getName());
184         // user code begin {1}
            // user code end
186      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
188         // user code end
            handleException(ivjExc);
190      }
         }
192      return ivjJPanel1;
      }
194   /**
       * Return the JPanel1FlowLayout property value.
196    * @return java.awt.FlowLayout
       */
198   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private java.awt.FlowLayout getJPanel1FlowLayout() {
200      java.awt.FlowLayout ivjJPanel1FlowLayout = null;
         try {
202         /* Create part */
            ivjJPanel1FlowLayout = new java.awt.FlowLayout();
204         ivjJPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
         } catch (java.lang.Throwable ivjExc) {
206         handleException(ivjExc);
         };
208      return ivjJPanel1FlowLayout;
      }
210   /**
       * Return the OKButton property value.
212    * @return javax.swing.JButton
       */
214   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getOKButton() {
216      if (ivjOKButton == null) {
            try {
218            ivjOKButton = new javax.swing.JButton();
               ivjOKButton.setName("OKButton");
220            ivjOKButton.setText("OK");
               // user code begin {1}
222            ivjOKButton.addActionListener( new OKButtonHandler() );
               // user code end
224         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
226            // user code end
               handleException(ivjExc);
228         }
         }
230      return ivjOKButton;
      }
232   /**
       * Return the ProtocolComboBox property value.
234    * @return javax.swing.JComboBox
       */
236   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JComboBox getProtocolComboBox() {
238      if (ivjProtocolComboBox == null) {
            try {
240            ivjProtocolComboBox = new javax.swing.JComboBox();
               ivjProtocolComboBox.setName("ProtocolComboBox");
242            // user code begin {1}
               // user code end
244         } catch (java.lang.Throwable ivjExc) {
```

```
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjProtocolComboBox;
   }
   /**
    * Return the ProtocolLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getProtocolLabel() {
      if (ivjProtocolLabel == null) {
         try {
            ivjProtocolLabel = new javax.swing.JLabel();
            ivjProtocolLabel.setName("ProtocolLabel");
            ivjProtocolLabel.setText("Connection:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjProtocolLabel;
   }
   /**
    * Return the ServerNameField property value.
    * @return javax.swing.JTextField
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextField getServerNameField() {
      if (ivjServerNameField == null) {
         try {
            ivjServerNameField = new javax.swing.JTextField();
            ivjServerNameField.setName("ServerNameField");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjServerNameField;
   }
   /**
    * Return the ServerNameLabel property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getServerNameLabel() {
      if (ivjServerNameLabel == null) {
         try {
            ivjServerNameLabel = new javax.swing.JLabel();
            ivjServerNameLabel.setName("ServerNameLabel");
            ivjServerNameLabel.setText("Server␣name:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
```

```
      }
312   return ivjServerNameLabel;
    }
314 /**
     * Return the JTextField1 property value.
316  * @return javax.swing.JTextField
     */
318 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JTextField getServerPortField() {
320   if (ivjServerPortField == null) {
        try {
322        ivjServerPortField = new javax.swing.JTextField();
           ivjServerPortField.setName("ServerPortField");
324        // user code begin {1}
           // user code end
326     } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
328        // user code end
           handleException(ivjExc);
330     }
      }
332   return ivjServerPortField;
    }
334 /**
     * Return the ServerPortLabel property value.
336  * @return javax.swing.JLabel
     */
338 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getServerPortLabel() {
340   if (ivjServerPortLabel == null) {
        try {
342        ivjServerPortLabel = new javax.swing.JLabel();
           ivjServerPortLabel.setName("ServerPortLabel");
344        ivjServerPortLabel.setText("Server␣port:");
           // user code begin {1}
346        // user code end
        } catch (java.lang.Throwable ivjExc) {
348        // user code begin {2}
           // user code end
350        handleException(ivjExc);
        }
352   }
      return ivjServerPortLabel;
354 }
    /**
356  * Return the JPanel2 property value.
     * @return javax.swing.JPanel
358  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
360 private javax.swing.JPanel getTopPanel() {
      if (ivjTopPanel == null) {
362     try {
           ivjTopPanel = new javax.swing.JPanel();
364        ivjTopPanel.setName("TopPanel");
           ivjTopPanel.setLayout(new java.awt.GridBagLayout());
366
           java.awt.GridBagConstraints constraintsServerNameLabel = new java.awt.GridBagConstraints();
368        constraintsServerNameLabel.gridx = 0; constraintsServerNameLabel.gridy = 0;
           constraintsServerNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
370        constraintsServerNameLabel.weighty = 0.2;
           constraintsServerNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
372        getTopPanel().add(getServerNameLabel(), constraintsServerNameLabel);

374        java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
           constraintsServerNameField.gridx = 1; constraintsServerNameField.gridy = 0;
376        constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
```

```
            constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHEAST;
378         constraintsServerNameField.weightx = 1.0;
            constraintsServerNameField.weighty = 0.2;
380         constraintsServerNameField.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getServerNameField(), constraintsServerNameField);
382
            java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
384         constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
            constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
386         constraintsServerPortLabel.weighty = 0.2;
            constraintsServerPortLabel.insets = new java.awt.Insets(6, 6, 6, 6);
388         getTopPanel().add(getServerPortLabel(), constraintsServerPortLabel);

390         java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
            constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
392         constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsServerPortField.weightx = 1.0;
394         constraintsServerPortField.weighty = 0.2;
            constraintsServerPortField.ipadx = 100;
396         constraintsServerPortField.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getServerPortField(), constraintsServerPortField);
398
            java.awt.GridBagConstraints constraintsProtocolLabel = new java.awt.GridBagConstraints();
400         constraintsProtocolLabel.gridx = 0; constraintsProtocolLabel.gridy = 2;
            constraintsProtocolLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
402         constraintsProtocolLabel.weighty = 0.2;
            constraintsProtocolLabel.insets = new java.awt.Insets(6, 6, 6, 6);
404         getTopPanel().add(getProtocolLabel(), constraintsProtocolLabel);

406         java.awt.GridBagConstraints constraintsProtocolComboBox = new java.awt.GridBagConstraints();
            constraintsProtocolComboBox.gridx = 1; constraintsProtocolComboBox.gridy = 2;
408         constraintsProtocolComboBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsProtocolComboBox.weightx = 1.0;
410         constraintsProtocolComboBox.weighty = 0.2;
            constraintsProtocolComboBox.insets = new java.awt.Insets(6, 6, 6, 6);
412         getTopPanel().add(getProtocolComboBox(), constraintsProtocolComboBox);
            // user code begin {1}
414         Border matteBorder = BorderFactory.createMatteBorder(6,6,6,6,ivjTopPanel.getBackground());
            Border lineBorder = BorderFactory.createLineBorder(Color.gray);
416         Border compoundBorder = BorderFactory.createCompoundBorder(lineBorder, matteBorder);
            Border secondCompoundBorder = BorderFactory.createCompoundBorder(matteBorder, compoundBorder);
418         ivjTopPanel.setBorder(secondCompoundBorder);
            // user code end
420       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
422         // user code end
            handleException(ivjExc);
424       }
        }
426     return ivjTopPanel;
    }
428 /**
     * Called whenever the part throws an exception.
430  * @param exception java.lang.Throwable
     */
432 private void handleException(java.lang.Throwable exception) {

434     /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
436     // exception.printStackTrace(System.out);
    }
438 /**
     * Initialize the class.
440  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
442 private void initialize() {
```

```
      try {
         // user code begin {1}
         // user code end
         setName("ServerSetup");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         setSize(458, 365);
         setTitle("Server␣setup");
         setContentPane(getJDialogContentPane());
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      // user code begin {2}
      setLocationRelativeTo( getOwner() );
      // user code end
   }
   /**
    * main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
    */
   public static void main(java.lang.String[] args) {
      try {
         ServerSetup aServerSetup;
         aServerSetup = new ServerSetup();
         aServerSetup.setModal(true);
         aServerSetup.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
            };
         });
         aServerSetup.setVisible(true);
      } catch (Throwable exception) {
         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
      }
   }
   /**
    * Insert the method's description here.
    * Creation date: (05-08-00 20:33:03)
    */
   public void show()
   {
      getServerNameField().setText( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME
           () ));
      getServerPortField().setText( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_PORT
           () ));
      if ( getProtocolComboBox().getItemCount() == 0 )
      {
         for ( int i = 0; i < ServerConstants.getCONNECTION_TYPES().length; i++ )
         {
            getProtocolComboBox().addItem( ServerConstants.getCONNECTION_TYPES()[i] );
         }
      }
      super.show();
   }
   /**
    * Insert the method's description here.
    * Creation date: (05-08-00 20:54:36)
    */
   public void submitAndClose()
   {
      ServerInfo.getPreferences().setProperty( ServerConstants.getSERVER_NAME(), getServerNameField().
           getText() );
      ServerInfo.getPreferences().setProperty( ServerConstants.getSERVER_PORT(), getServerPortField().
           getText() );
      ServerInfo.getPreferences().setProperty( ServerConstants.getCONNECTION_TYPE(), (String)
           getProtocolComboBox().getSelectedItem() );
```

```
504
      ServerInfo.writePreferences();
506   dispose();

508 }
    }
```

Listing C.34: AboutDialog.java

```
    package peerviewmisc;
 2
    import javax.swing.*;
 4  import javax.swing.border.*;
    import javax.swing.table.*;
 6  import javax.swing.event.*;

 8  /**
     * The PeerView about dialog box class
10   * Creation date: (26-05-00 08:56:28)
     * @author:
12   */
    public class AboutDialog extends JDialog {
14     private JPanel ivjJDialogContentPane = null;
       private JLabel ivjLogoLabel = null;
16     private JLabel ivjRulerLabel = null;
       private JLabel ivjTextLabel = null;
18  /**
     * AboutDialog constructor comment.
20   */
    public AboutDialog() {
22     super();
       initialize();
24  }
    /**
26   * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
28   */
    public AboutDialog(java.awt.Dialog owner) {
30     super(owner);
    }
32  /**
     * AboutDialog constructor comment.
34   * @param owner java.awt.Dialog
     * @param title java.lang.String
36   */
    public AboutDialog(java.awt.Dialog owner, String title) {
38     super(owner, title);
    }
40  /**
     * AboutDialog constructor comment.
42   * @param owner java.awt.Dialog
     * @param title java.lang.String
44   * @param modal boolean
     */
46  public AboutDialog(java.awt.Dialog owner, String title, boolean modal) {
       super(owner, title, modal);
48  }
    /**
50   * AboutDialog constructor comment.
     * @param owner java.awt.Dialog
52   * @param modal boolean
     */
54  public AboutDialog(java.awt.Dialog owner, boolean modal) {
       super(owner, modal);
56  }
    /**
58   * AboutDialog constructor comment.
```

```
     * @param owner java.awt.Frame
 60   */
     public AboutDialog(java.awt.Frame owner) {
 62     super(owner);
     }
 64  /**
     * AboutDialog constructor comment.
 66   * @param owner java.awt.Frame
     * @param title java.lang.String
 68   */
     public AboutDialog(java.awt.Frame owner, String title) {
 70     super(owner, title);
     }
 72  /**
     * AboutDialog constructor comment.
 74   * @param owner java.awt.Frame
     * @param title java.lang.String
 76   * @param modal boolean
     */
 78  public AboutDialog(java.awt.Frame owner, String title, boolean modal) {
       super(owner, title, modal);
 80  }
     /**
 82   * AboutDialog constructor comment.
     * @param owner java.awt.Frame
 84   * @param modal boolean
     */
 86  public AboutDialog(java.awt.Frame owner, boolean modal) {
       super(owner, modal);
 88  }
     /**
 90   * Return the JDialogContentPane property value.
     * @return javax.swing.JPanel
 92   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
 94  private javax.swing.JPanel getJDialogContentPane() {
       if (ivjJDialogContentPane == null) {
 96      try {
           ivjJDialogContentPane = new javax.swing.JPanel();
 98        ivjJDialogContentPane.setName("JDialogContentPane");
           ivjJDialogContentPane.setLayout(new java.awt.GridBagLayout());
100        ivjJDialogContentPane.setBackground(java.awt.Color.white);

102        java.awt.GridBagConstraints constraintsLogoLabel = new java.awt.GridBagConstraints();
           constraintsLogoLabel.gridx = 0; constraintsLogoLabel.gridy = 0;
104        constraintsLogoLabel.gridwidth = 2;
           constraintsLogoLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
106        constraintsLogoLabel.weightx = 1.0;
           constraintsLogoLabel.ipady = 5;
108        constraintsLogoLabel.insets = new java.awt.Insets(5, 5, 0, 5);
           getJDialogContentPane().add(getLogoLabel(), constraintsLogoLabel);
110
           java.awt.GridBagConstraints constraintsRulerLabel = new java.awt.GridBagConstraints();
112        constraintsRulerLabel.gridx = 0; constraintsRulerLabel.gridy = 1;
           constraintsRulerLabel.gridwidth = 3;
114        constraintsRulerLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
           constraintsRulerLabel.ipady = 10;
116        constraintsRulerLabel.insets = new java.awt.Insets(0, 5, 5, 5);
           getJDialogContentPane().add(getRulerLabel(), constraintsRulerLabel);
118
           java.awt.GridBagConstraints constraintsTextLabel = new java.awt.GridBagConstraints();
120        constraintsTextLabel.gridx = 0; constraintsTextLabel.gridy = 2;
           constraintsTextLabel.gridwidth = 2;
122        constraintsTextLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
           constraintsTextLabel.weightx = 1.0;
124        constraintsTextLabel.ipady = 10;
```

```
            constraintsTextLabel.insets = new java.awt.Insets(5, 5, 5, 5);
126         getJDialogContentPane().add(getTextLabel(), constraintsTextLabel);
            // user code begin {1}
128         // user code end
         } catch (java.lang.Throwable ivjExc) {
130         // user code begin {2}
            // user code end
132         handleException(ivjExc);
         }
134   }
      return ivjJDialogContentPane;
136 }
    /**
138  * Return the LogoLabel property value.
     * @return javax.swing.JLabel
140  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
142 private javax.swing.JLabel getLogoLabel() {
      if (ivjLogoLabel == null) {
144      try {
            ivjLogoLabel = new javax.swing.JLabel();
146         ivjLogoLabel.setName("LogoLabel");
            ivjLogoLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/peerviewlogo2.gif")));
148         ivjLogoLabel.setText("");
            ivjLogoLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
150         ivjLogoLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            // user code begin {1}
152   /*    Border raisedBorder = BorderFactory.createEtchedBorder( new java.awt.Color( 33, 132, 132 ), new
         java.awt.Color( 189, 189, 189 ));

154         Border matteBorder = BorderFactory.createMatteBorder(6,6,6,6,ivjLogoLabel.getBackground() );
            Border compoundBorder = BorderFactory.createCompoundBorder(matteBorder, raisedBorder);
156         ivjLogoLabel.setBorder(compoundBorder);
      */    ivjLogoLabel.setBackground( java.awt.Color.white );
158         ivjLogoLabel.setForeground( java.awt.Color.white );
            ivjLogoLabel.setVerticalTextPosition( SwingConstants.BOTTOM );
160      // ivjLogoLabel.setVerticalAlignment( SwingConstants.BOTTOM );
            ivjLogoLabel.setText( "Release␣0.900␣(beta)" );
162
            // user code end
164      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
166         // user code end
            handleException(ivjExc);
168      }
      }
170   return ivjLogoLabel;
    }
172 /**
     * Return the RulerLabel property value.
174  * @return javax.swing.JLabel
     */
176 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getRulerLabel() {
178   if (ivjRulerLabel == null) {
         try {
180         ivjRulerLabel = new javax.swing.JLabel();
            ivjRulerLabel.setName("RulerLabel");
182         ivjRulerLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ruler2.gif")));
            ivjRulerLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
184         ivjRulerLabel.setText("");
            ivjRulerLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
186         // user code begin {1}
            // user code end
188      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
```

```
190         // user code end
            handleException(ivjExc);
192      }
         }
194      return ivjRulerLabel;
      }
196   /**
       * Return the TextLabel property value.
198    * @return javax.swing.JLabel
       */
200   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JLabel getTextLabel() {
202      if (ivjTextLabel == null) {
            try {
204            ivjTextLabel = new javax.swing.JLabel();
               ivjTextLabel.setName("TextLabel");
206            ivjTextLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/text.gif")));
               ivjTextLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
208            ivjTextLabel.setText("");
               ivjTextLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
210            // user code begin {1}
               // user code end
212         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
214            // user code end
               handleException(ivjExc);
216         }
         }
218      return ivjTextLabel;
      }
220   /**
       * Called whenever the part throws an exception.
222    * @param exception java.lang.Throwable
       */
224   private void handleException(java.lang.Throwable exception) {

226      /* Uncomment the following lines to print uncaught exceptions to stdout */
         // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
228      // exception.printStackTrace(System.out);
      }
230   /**
       * Initialize the class.
232    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
234   private void initialize() {
         try {
236         // user code begin {1}
            // user code end
238         setName("AboutDialog");
            setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
240         setTitle("About␣PeerView");
            setBackground(java.awt.SystemColor.window);
242         setModal(true);
            setSize(483, 267);
244         setResizable(false);
            setContentPane(getJDialogContentPane());
246      } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
248      }
         // user code begin {2}
250      setLocationRelativeTo( getOwner() );
         // user code end
252   }
      /**
254    * main entrypoint - starts the part when it is run as an application
       * @param args java.lang.String[]
```

```
 */
public static void main(java.lang.String[] args) {
   try {
      AboutDialog aAboutDialog;
      aAboutDialog = new AboutDialog();
      aAboutDialog.setModal(true);
      aAboutDialog.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aAboutDialog.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
}
```

Listing C.35: CompressedDocumentPackage.java

```
package peerviewmisc;

 import java.io.Serializable;

/**
 * This class implements a container for transmitting compressed byte arrays and their descriptors
     around a network using
 * the Java serialization mechanism for marshalling and unmarshalling.
 * Creation date: (19-08-00 13:07:49)
 * @author:
 */
public class CompressedDocumentPackage implements Serializable
{
   private int uncompressedLength;
   private int compressedLength;
   private byte[] compressedContents = null;
   private java.lang.String path = null;
   private java.lang.String name = null;
/**
 * CompressedDocumentPackage constructor comment.
 */
public CompressedDocumentPackage() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (19-08-00 13:10:51)
 * @return byte[]
 */
public byte[] getCompressedContents() {
   return compressedContents;
}
/**
 * Insert the method's description here.
 * Creation date: (19-08-00 13:08:45)
 * @return int
 */
public int getCompressedLength() {
   return compressedLength;
}
/**
 * Insert the method's description here.
 * Creation date: (28-10-00 00:18:50)
 * @return java.lang.String
 */
public java.lang.String getName() {
```

```
    return name;
  }
  /**
   * Insert the method's description here.
   * Creation date: (05-09-00 14:24:28)
   * @return java.lang.String
   */
  public java.lang.String getPath() {
    return path;
  }
  /**
   * Insert the method's description here.
   * Creation date: (19-08-00 13:08:13)
   * @return int
   */
  public int getUncompressedLength() {
    return uncompressedLength;
  }
  /**
   * Insert the method's description here.
   * Creation date: (19-08-00 13:10:51)
   * @param newCompressedContents byte[]
   */
  public void setCompressedContents(byte[] newCompressedContents) {
    compressedContents = newCompressedContents;
  }
  /**
   * Insert the method's description here.
   * Creation date: (19-08-00 13:08:45)
   * @param newCompressedLength int
   */
  public void setCompressedLength(int newCompressedLength) {
    compressedLength = newCompressedLength;
  }
  /**
   * Insert the method's description here.
   * Creation date: (28-10-00 00:18:50)
   * @param newName java.lang.String
   */
  public void setName(java.lang.String newName) {
    name = newName;
  }
  /**
   * Insert the method's description here.
   * Creation date: (05-09-00 14:24:28)
   * @param newPath java.lang.String
   */
  public void setPath(java.lang.String newPath) {
    path = newPath;
  }
  /**
   * Insert the method's description here.
   * Creation date: (19-08-00 13:08:13)
   * @param newUncompressedLength int
   */
  public void setUncompressedLength(int newUncompressedLength) {
    uncompressedLength = newUncompressedLength;
  }
  }
```

Listing C.36: ConfirmationBox.java

```
package peerviewmisc;

 import javax.swing.event.*;
 import java.awt.event.*;
```

```
/**
 * This class implements the confirmation box displayed when a user is asked to confirm or deny an
     action.
 * Creation date: (12-07-00 11:11:22)
 * @author:
 */

public class ConfirmationBox extends javax.swing.JDialog {

   public class OKButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         submitAndClose();
      }
   }

   public class CancelButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         cancelAndClose();
      }
   }
   private javax.swing.JButton ivjCancelButton = null;
   private javax.swing.JPanel ivjJDialogContentPane = null;
   private javax.swing.JButton ivjOKButton = null;
   private boolean confirmed;
   private javax.swing.JLabel ivjIconLabel = null;
   private javax.swing.JPanel ivjMainPanel = null;
   private javax.swing.JTextPane ivjMessagePanel = null;
   private javax.swing.JPanel ivjButtonPanel = null;
   private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
/**
 * ConfirmationBox constructor comment.
 */
public ConfirmationBox() {
   super();
   initialize();
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Dialog
 */
public ConfirmationBox(java.awt.Dialog owner) {
   super(owner);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public ConfirmationBox(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * ConfirmationBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public ConfirmationBox(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * ConfirmationBox constructor comment.
```

```
     * @param owner java.awt.Dialog
 72  * @param modal boolean
     */
 74  public ConfirmationBox(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
 76  }
     /**
 78  * ConfirmationBox constructor comment.
     * @param owner java.awt.Frame
 80  */
     public ConfirmationBox(java.awt.Frame owner) {
 82     super(owner);
     }
 84  /**
     * ConfirmationBox constructor comment.
 86  * @param owner java.awt.Frame
     * @param title java.lang.String
 88  */
     public ConfirmationBox(java.awt.Frame owner, String title) {
 90     super(owner, title);
     }
 92  /**
     * ConfirmationBox constructor comment.
 94  * @param owner java.awt.Frame
     * @param title java.lang.String
 96  * @param modal boolean
     */
 98  public ConfirmationBox(java.awt.Frame owner, String title, boolean modal) {
        super(owner, title, modal);
100  }
     /**
102  * ConfirmationBox constructor comment.
     * @param owner java.awt.Frame
104  * @param modal boolean
     */
106  public ConfirmationBox(java.awt.Frame owner, boolean modal) {
        super(owner, modal);
108  }
     /**
110  * Insert the method's description here.
     * Creation date: (12-07-00 11:37:04)
112  */
     public void cancelAndClose()
114  {
        setConfirmed( false );
116     dispose();
     }
118  /**
     * Return the JPanel1 property value.
120  * @return javax.swing.JPanel
     */
122  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getButtonPanel() {
124     if (ivjButtonPanel == null) {
           try {
126           ivjButtonPanel = new javax.swing.JPanel();
              ivjButtonPanel.setName("ButtonPanel");
128           ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
              getButtonPanel().add(getOKButton(), getOKButton().getName());
130           getButtonPanel().add(getCancelButton(), getCancelButton().getName());
              // user code begin {1}
132           // user code end
           } catch (java.lang.Throwable ivjExc) {
134           // user code begin {2}
              // user code end
136           handleException(ivjExc);
```

```
        }
    }
    return ivjButtonPanel;
}
/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getButtonPanelFlowLayout() {
    java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
    try {
        /* Create part */
        ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    };
    return ivjButtonPanelFlowLayout;
}
/**
 * Return the CancelButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCancelButton() {
    if (ivjCancelButton == null) {
        try {
            ivjCancelButton = new javax.swing.JButton();
            ivjCancelButton.setName("CancelButton");
            ivjCancelButton.setText("Cancel");
            // user code begin {1}
            ivjCancelButton.addActionListener( new CancelButtonHandler() );
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjCancelButton;
}
/**
 * Return the IconLabel property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JLabel getIconLabel() {
    if (ivjIconLabel == null) {
        try {
            ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
            ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                general/ContextualHelp24.gif")));
            ivjIconLabel.setText("");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjIconLabel;
}
/**
```

```
202   * Return the JDialogContentPane property value.
      * @return javax.swing.JPanel
204   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
206 private javax.swing.JPanel getJDialogContentPane() {
       if (ivjJDialogContentPane == null) {
208       try {
             ivjJDialogContentPane = new javax.swing.JPanel();
210          ivjJDialogContentPane.setName("JDialogContentPane");
             ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
212          getJDialogContentPane().add(getButtonPanel(), "South");
             getJDialogContentPane().add(getMainPanel(), "Center");
214          // user code begin {1}
             // user code end
216       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
218          // user code end
             handleException(ivjExc);
220       }
       }
222    return ivjJDialogContentPane;
    }
224 /**
      * Return the JPanel2 property value.
226   * @return javax.swing.JPanel
      */
228 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getMainPanel() {
230    if (ivjMainPanel == null) {
       try {
232          ivjMainPanel = new javax.swing.JPanel();
             ivjMainPanel.setName("MainPanel");
234          ivjMainPanel.setLayout(new java.awt.GridBagLayout());

236          java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
             constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
238          constraintsMessagePanel.fill = java.awt.GridBagConstraints.NONE;
             constraintsMessagePanel.weightx = 1.0;
240          constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
             getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
242
             java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
244          constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
             constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
246          getMainPanel().add(getIconLabel(), constraintsIconLabel);
             // user code begin {1}
248          getMainPanel().setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
             // user code end
250       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
252          // user code end
             handleException(ivjExc);
254       }
       }
256    return ivjMainPanel;
    }
258 /**
      * Return the JTextPane1 property value.
260   * @return javax.swing.JTextPane
      */
262 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JTextPane getMessagePanel() {
264    if (ivjMessagePanel == null) {
       try {
266          ivjMessagePanel = new javax.swing.JTextPane();
             ivjMessagePanel.setName("MessagePanel");
```

```
268         ivjMessagePanel.setEditable(false);
            // user code begin {1}
270         ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
            // user code end
272      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
274         // user code end
            handleException(ivjExc);
276      }
         }
278      return ivjMessagePanel;
      }
280   /**
       * Return the OKButton property value.
282    * @return javax.swing.JButton
       */
284   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getOKButton() {
286      if (ivjOKButton == null) {
            try {
288         ivjOKButton = new javax.swing.JButton();
            ivjOKButton.setName("OKButton");
290         ivjOKButton.setMnemonic('O');
            ivjOKButton.setText("OK");
292         // user code begin {1}
            ivjOKButton.addActionListener( new OKButtonHandler() );
294         // user code end
            } catch (java.lang.Throwable ivjExc) {
296         // user code begin {2}
            // user code end
298         handleException(ivjExc);
            }
300      }
         return ivjOKButton;
302   }
      /**
304    * Called whenever the part throws an exception.
       * @param exception java.lang.Throwable
306    */
      private void handleException(java.lang.Throwable exception) {
308
         /* Uncomment the following lines to print uncaught exceptions to stdout */
310      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
         // exception.printStackTrace(System.out);
312   }
      /**
314    * Initialize the class.
       */
316   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private void initialize() {
318      try {
            // user code begin {1}
320         // user code end
            setName("ConfirmationBox");
322         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            setSize(436, 181);
324         setEnabled(true);
            setModal(true);
326         setTitle("Confirmation");
            setContentPane(getJDialogContentPane());
328      } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
330      }
         // user code begin {2}
332
         // user code end
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (12-07-00 11:26:11)
 * @return boolean
 */
public boolean isConfirmed() {
   return confirmed;
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      ConfirmationBox aConfirmationBox;
      aConfirmationBox = new ConfirmationBox();
      aConfirmationBox.setModal(true);
      aConfirmationBox.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aConfirmationBox.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (12-07-00 11:26:11)
 * @param newConfirmed boolean
 */
public void setConfirmed(boolean newConfirmed) {
   confirmed = newConfirmed;
}
/**
 * Insert the method's description here.
 * Creation date: (12-07-00 11:44:14)
 * @param newText java.lang.String
 */
public void setText(String newText)
{
   getMessagePanel().setText( newText );
   getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel
        ().getFontMetrics( getMessagePanel().getFont() )));
   getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
   pack();
   setLocationRelativeTo( getOwner() );
}
/**
 * Insert the method's description here.
 * Creation date: (12-07-00 11:36:53)
 */
public void submitAndClose()
{
   setConfirmed( true );
   dispose();
}
}
```

Listing C.37: DataPackage.java

```
package peerviewmisc;

 import java.io.*;
```

```
 import java.util.Hashtable;

 /**
 * This class implements a container used for transmitting data between clients.
 * Creation date: (04-07-00 17:10:18)
 * @author:
 */

public class DataPackage implements Serializable
{
   private final static java.lang.String AUTHOR_NAME = "Author␣name";
   private final static java.lang.String DOCPATH = "Document␣name␣and␣path";
   private final static java.lang.String DOCNAME = "Document␣name␣(without␣path)";
   private final static java.lang.String DESCRIPTION = "Description";
   private java.util.Properties properties = new java.util.Properties();
   private java.lang.Object contents = null;
   private final static java.lang.String WORKGROUP_ID = "Workgroup␣ID";
   private final static java.lang.String MESSAGE_ID = "Message␣ID";
   private final static java.lang.String CLIENT_NAME = "Client␣name";
/**
 * DataPackage constructor comment.
 */
public DataPackage() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:45:37)
 * @return java.lang.String
 */
public final static java.lang.String getAUTHOR_NAME() {
   return AUTHOR_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (29-10-00 19:05:42)
 * @return java.lang.String
 */
public final static java.lang.String getCLIENT_NAME() {
   return CLIENT_NAME;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:59:23)
 * @return java.lang.Object
 */
public java.lang.Object getContents() {
   return contents;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:52:15)
 * @return java.lang.String
 */
public final static java.lang.String getDESCRIPTION() {
   return DESCRIPTION;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:49:36)
 * @return java.lang.String
 */
public final static java.lang.String getDOCNAME() {
   return DOCNAME;
}
/**
```

```
 * Insert the method's description here.
 * Creation date: (27-10-00 22:49:02)
 * @return java.lang.String
 */
public final static java.lang.String getDOCPATH() {
   return DOCPATH;
}
/**
 * Insert the method's description here.
 * Creation date: (28-10-00 00:03:49)
 * @return java.lang.String
 */
public final static java.lang.String getMESSAGE_ID() {
   return MESSAGE_ID;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:55:47)
 * @return java.util.Properties
 */
public java.util.Properties getProperties() {
   return properties;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 23:18:45)
 * @return java.lang.String
 */
public final static java.lang.String getWORKGROUP_ID() {
   return WORKGROUP_ID;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:59:23)
 * @param newContents java.lang.Object
 */
public void setContents(java.lang.Object newContents) {
   contents = newContents;
}
/**
 * Insert the method's description here.
 * Creation date: (27-10-00 22:55:47)
 * @param newProperties java.util.Properties
 */
public void setProperties(java.util.Properties newProperties) {
   properties = newProperties;
}
}
```

Listing C.38: DeleteGroup.java

```
package peerviewmisc;

/**
 * This class implements the delete group dialog box.
 * Creation date: (29-06-00 16:19:12)
 * @author:
 */
public class DeleteGroup extends DisplayGroup {
/**
 * DeleteGroup constructor comment.
 */
public DeleteGroup() {
   super();
   initialize();
}
/**
```

```
     * DeleteGroup constructor comment.
18   * @param owner java.awt.Dialog
     */
20  public DeleteGroup(java.awt.Dialog owner) {
       super(owner);
22  }
    /**
24   * DeleteGroup constructor comment.
     * @param owner java.awt.Dialog
26   * @param title java.lang.String
     */
28  public DeleteGroup(java.awt.Dialog owner, String title) {
       super(owner, title);
30  }
    /**
32   * DeleteGroup constructor comment.
     * @param owner java.awt.Dialog
34   * @param title java.lang.String
     * @param modal boolean
36   */
    public DeleteGroup(java.awt.Dialog owner, String title, boolean modal) {
38     super(owner, title, modal);
    }
40  /**
     * DeleteGroup constructor comment.
42   * @param owner java.awt.Dialog
     * @param modal boolean
44   */
    public DeleteGroup(java.awt.Dialog owner, boolean modal) {
46     super(owner, modal);
    }
48  /**
     * DeleteGroup constructor comment.
50   * @param owner java.awt.Frame
     */
52  public DeleteGroup(java.awt.Frame owner) {
       super(owner);
54  }
    /**
56   * DeleteGroup constructor comment.
     * @param owner java.awt.Frame
58   * @param title java.lang.String
     */
60  public DeleteGroup(java.awt.Frame owner, String title) {
       super(owner, title);
62  }
    /**
64   * DeleteGroup constructor comment.
     * @param owner java.awt.Frame
66   * @param title java.lang.String
     * @param modal boolean
68   */
    public DeleteGroup(java.awt.Frame owner, String title, boolean modal) {
70     super(owner, title, modal);
    }
72  /**
     * DeleteGroup constructor comment.
74   * @param owner java.awt.Frame
     * @param modal boolean
76   */
    public DeleteGroup(java.awt.Frame owner, boolean modal) {
78     super(owner, modal);
    }
80  /**
     * Called whenever the part throws an exception.
82   * @param exception java.lang.Throwable
```

```
 */
private void handleException(java.lang.Throwable exception) {

  /* Uncomment the following lines to print uncaught exceptions to stdout */
  // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
  // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
  try {
    // user code begin {1}
    // user code end
    setName("DeleteGroup");
  } catch (java.lang.Throwable ivjExc) {
    handleException(ivjExc);
  }
  // user code begin {2}
  setTitle( "Delete␣group" );
  editable( false );
  getOKButton().setText( "Delete" );
  // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
  try {
    DeleteGroup aDeleteGroup;
    aDeleteGroup = new DeleteGroup();
    aDeleteGroup.setModal(true);
    aDeleteGroup.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(java.awt.event.WindowEvent e) {
        System.exit(0);
      };
    });
    aDeleteGroup.setVisible(true);
  } catch (Throwable exception) {
    System.err.println("Exception␣occurred␣in␣main()␣of␣serverapp.DeleteGroup");
    exception.printStackTrace(System.out);
  }
}
/**
 * Insert the method's description here.
 * Creation date: (27-07-00 09:42:39)
 */
public void submitAndClose() {}
}
```

Listing C.39: DisplayGroup.java

```
package peerviewmisc;

 import javax.swing.JComponent;
 import javax.swing.JTextField;
 import javax.swing.text.JTextComponent;
 import java.awt.Component;
 import javax.swing.event.*;
 import java.awt.event.*;

 /**
 * This abstract class is the root of the class hierarchy of dialog boxes that accept input from the
     user.
 * Creation date: (29-06-00 10:21:35)
```

```
 * @author:
 */
public abstract class DisplayGroup extends javax.swing.JDialog {

   public class OKButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         submitAndClose();
      }
   }

   public class CancelButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         cancelAndClose();
      }
   }
   protected javax.swing.JButton ivjCancelButton = null;
   protected javax.swing.JLabel ivjDescriptionLabel = null;
   protected javax.swing.JLabel ivjGroupNameLabel = null;
   protected javax.swing.JPanel ivjJDialogContentPane = null;
   protected javax.swing.JButton ivjOKButton = null;
   protected javax.swing.JPanel ivjButtonPanel = null;
   protected java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
   protected javax.swing.JPanel ivjMainPanel = null;
   protected WorkGroup workGroup = null;
   protected javax.swing.JTextPane ivjDescriptionField = null;
   protected javax.swing.JTextField ivjGroupNameField = null;
   protected clientapp.GroupDirectory groupDirectory;
   protected boolean isDiscarded;
   protected javax.swing.JScrollPane ivjDescriptionScrollPane = null;
/**
 * DisplayGroup constructor comment.
 */
public DisplayGroup() {
   super();
   initialize();
}
/**
 * DisplayGroup constructor comment.
 * @param owner java.awt.Dialog
 */
public DisplayGroup(java.awt.Dialog owner) {
   super(owner);
}
/**
 * DisplayGroup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public DisplayGroup(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * DisplayGroup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public DisplayGroup(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * DisplayGroup constructor comment.
```

```
     * @param owner java.awt.Dialog
 80   * @param modal boolean
     */
 82  public DisplayGroup(java.awt.Dialog owner, boolean modal) {
        super(owner, modal);
 84  }
     /**
 86   * DisplayGroup constructor comment.
     * @param owner java.awt.Frame
 88   */
     public DisplayGroup(java.awt.Frame owner) {
 90     super(owner);
     }
 92  /**
     * DisplayGroup constructor comment.
 94   * @param owner java.awt.Frame
     * @param title java.lang.String
 96   */
     public DisplayGroup(java.awt.Frame owner, String title) {
 98     super(owner, title);
     }
100  /**
     * DisplayGroup constructor comment.
102   * @param owner java.awt.Frame
     * @param title java.lang.String
104   * @param modal boolean
     */
106  public DisplayGroup(java.awt.Frame owner, String title, boolean modal) {
        super(owner, title, modal);
108  }
     /**
110   * DisplayGroup constructor comment.
     * @param owner java.awt.Frame
112   * @param modal boolean
     */
114  public DisplayGroup(java.awt.Frame owner, boolean modal) {
        super(owner, modal);
116  }
     /**
118   * Insert the method's description here.
     * Creation date: (26-07-00 19:28:46)
120   */
     public void cancelAndClose()
122  {
        setIsDiscarded( true );
124     dispose();
     }
126  /**
     * Set editability of text fields in the main panel of the display group dialog box.
128   * Creation date: (29-06-00 15:26:57)
     * @param edit boolean
130   */
     public void editable(boolean edit)
132  {
        Component components[] = getMainPanel().getComponents();
134
        for (int i = 0; i < components.length; i++)
136     {
           if (components[i] instanceof JTextComponent)
138        {
              ((JTextComponent) components[i]).setEditable(edit);
140        }
        }
142  }
     /**
144   * Return the JPanel2 property value.
```

```
     * @return javax.swing.JPanel
146  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
148  private javax.swing.JPanel getButtonPanel() {
        if (ivjButtonPanel == null) {
150        try {
              ivjButtonPanel = new javax.swing.JPanel();
152           ivjButtonPanel.setName("ButtonPanel");
              ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
154           getButtonPanel().add(getOKButton(), getOKButton().getName());
              getButtonPanel().add(getCancelButton(), getCancelButton().getName());
156           // user code begin {1}
              // user code end
158        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
160           // user code end
              handleException(ivjExc);
162        }
        }
164     return ivjButtonPanel;
     }
166  /**
     * Return the ButtonPanelFlowLayout property value.
168  * @return java.awt.FlowLayout
     */
170  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private java.awt.FlowLayout getButtonPanelFlowLayout() {
172     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
        try {
174        /* Create part */
           ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
176        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
        } catch (java.lang.Throwable ivjExc) {
178        handleException(ivjExc);
        };
180     return ivjButtonPanelFlowLayout;
     }
182  /**
     * Return the CancelButton property value.
184  * @return javax.swing.JButton
     */
186  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     public javax.swing.JButton getCancelButton() {
188     if (ivjCancelButton == null) {
           try {
190           ivjCancelButton = new javax.swing.JButton();
              ivjCancelButton.setName("CancelButton");
192           ivjCancelButton.setText("Cancel");
              // user code begin {1}
194           ivjCancelButton.addActionListener( new CancelButtonHandler() );
              // user code end
196        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
198           // user code end
              handleException(ivjExc);
200        }
        }
202     return ivjCancelButton;
     }
204  /**
     * Return the JTextPane1 property value.
206  * @return javax.swing.JTextPane
     */
208  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JTextPane getDescriptionField() {
210     if (ivjDescriptionField == null) {
```

```
        try {
212        ivjDescriptionField = new javax.swing.JTextPane();
           ivjDescriptionField.setName("DescriptionField");
214        ivjDescriptionField.setBounds(0, 0, 312, 137);
           // user code begin {1}
216        // user code end
        } catch (java.lang.Throwable ivjExc) {
218        // user code begin {2}
           // user code end
220        handleException(ivjExc);
        }
222  }
     return ivjDescriptionField;
224 }
    /**
226  * Return the DescriptionLabel property value.
     * @return javax.swing.JLabel
228  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
230 private javax.swing.JLabel getDescriptionLabel() {
     if (ivjDescriptionLabel == null) {
232     try {
           ivjDescriptionLabel = new javax.swing.JLabel();
234        ivjDescriptionLabel.setName("DescriptionLabel");
           ivjDescriptionLabel.setText("Description");
236        // user code begin {1}
           // user code end
238     } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
240        // user code end
           handleException(ivjExc);
242     }
     }
244  return ivjDescriptionLabel;
    }
246 /**
     * Return the DescriptionScrollPane property value.
248  * @return javax.swing.JScrollPane
     */
250 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JScrollPane getDescriptionScrollPane() {
252  if (ivjDescriptionScrollPane == null) {
        try {
254        ivjDescriptionScrollPane = new javax.swing.JScrollPane();
           ivjDescriptionScrollPane.setName("DescriptionScrollPane");
256        getDescriptionScrollPane().setViewportView(getDescriptionField());
           // user code begin {1}
258        // user code end
        } catch (java.lang.Throwable ivjExc) {
260        // user code begin {2}
           // user code end
262        handleException(ivjExc);
        }
264  }
     return ivjDescriptionScrollPane;
266 }
    /**
268  * Insert the method's description here.
     * Creation date: (26-07-00 17:28:18)
270  * @return clientapp.GroupDirectory
     */
272 public clientapp.GroupDirectory getGroupDirectory() {
     return groupDirectory;
274 }
    /**
276  * Return the JTextField1 property value.
```

```
     * @return javax.swing.JTextField
 278  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
 280 private javax.swing.JTextField getGroupNameField() {
        if (ivjGroupNameField == null) {
 282       try {
           ivjGroupNameField = new javax.swing.JTextField();
 284       ivjGroupNameField.setName("GroupNameField");
           // user code begin {1}
 286       // user code end
        } catch (java.lang.Throwable ivjExc) {
 288       // user code begin {2}
           // user code end
 290       handleException(ivjExc);
        }
 292    }
        return ivjGroupNameField;
 294 }
     /**
 296  * Return the GroupNameLabel property value.
     * @return javax.swing.JLabel
 298  */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
 300 private javax.swing.JLabel getGroupNameLabel() {
        if (ivjGroupNameLabel == null) {
 302       try {
           ivjGroupNameLabel = new javax.swing.JLabel();
 304       ivjGroupNameLabel.setName("GroupNameLabel");
           ivjGroupNameLabel.setText("Group␣name");
 306       // user code begin {1}
           // user code end
 308       } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
 310       // user code end
           handleException(ivjExc);
 312       }
        }
 314    return ivjGroupNameLabel;
     }
 316 /**
     * Return the JDialogContentPane property value.
 318  * @return javax.swing.JPanel
     */
 320 /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJDialogContentPane() {
 322    if (ivjJDialogContentPane == null) {
        try {
 324       ivjJDialogContentPane = new javax.swing.JPanel();
           ivjJDialogContentPane.setName("JDialogContentPane");
 326       ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
           getJDialogContentPane().add(getMainPanel(), "Center");
 328       getJDialogContentPane().add(getButtonPanel(), "South");
           // user code begin {1}
 330       // user code end
        } catch (java.lang.Throwable ivjExc) {
 332       // user code begin {2}
           // user code end
 334       handleException(ivjExc);
        }
 336    }
        return ivjJDialogContentPane;
 338 }
     /**
 340  * Return the JPanel1 property value.
     * @return javax.swing.JPanel
 342  */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getMainPanel() {
   if (ivjMainPanel == null) {
      try {
         ivjMainPanel = new javax.swing.JPanel();
         ivjMainPanel.setName("MainPanel");
         ivjMainPanel.setLayout(new java.awt.GridBagLayout());

         java.awt.GridBagConstraints constraintsGroupNameLabel = new java.awt.GridBagConstraints();
         constraintsGroupNameLabel.gridx = 0; constraintsGroupNameLabel.gridy = 0;
         constraintsGroupNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsGroupNameLabel.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getGroupNameLabel(), constraintsGroupNameLabel);

         java.awt.GridBagConstraints constraintsDescriptionLabel = new java.awt.GridBagConstraints();
         constraintsDescriptionLabel.gridx = 0; constraintsDescriptionLabel.gridy = 2;
         constraintsDescriptionLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsDescriptionLabel.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getDescriptionLabel(), constraintsDescriptionLabel);

         java.awt.GridBagConstraints constraintsGroupNameField = new java.awt.GridBagConstraints();
         constraintsGroupNameField.gridx = 1; constraintsGroupNameField.gridy = 0;
         constraintsGroupNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
         constraintsGroupNameField.anchor = java.awt.GridBagConstraints.EAST;
         constraintsGroupNameField.weightx = 1.0;
         constraintsGroupNameField.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getGroupNameField(), constraintsGroupNameField);

         java.awt.GridBagConstraints constraintsDescriptionScrollPane = new java.awt.GridBagConstraints()
              ;
         constraintsDescriptionScrollPane.gridx = 1; constraintsDescriptionScrollPane.gridy = 2;
         constraintsDescriptionScrollPane.fill = java.awt.GridBagConstraints.BOTH;
         constraintsDescriptionScrollPane.weightx = 1.0;
         constraintsDescriptionScrollPane.weighty = 1.0;
         constraintsDescriptionScrollPane.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getDescriptionScrollPane(), constraintsDescriptionScrollPane);
         // user code begin {1}
         getMainPanel().setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjMainPanel;
}
/**
 * Return the OKButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public javax.swing.JButton getOKButton() {
   if (ivjOKButton == null) {
      try {
         ivjOKButton = new javax.swing.JButton();
         ivjOKButton.setName("OKButton");
         ivjOKButton.setMnemonic('o');
         ivjOKButton.setText("OK");
         // user code begin {1}
         ivjOKButton.addActionListener( new OKButtonHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
```

```
      }
   }
   return ivjOKButton;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 20:56:36)
 * @return peerviewmisc.WorkGroup
 */
public WorkGroup getWorkGroup() {
   return workGroup;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
      setName("DisplayGroup");
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setSize(426, 240);
      setContentPane(getJDialogContentPane());
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   }
   // user code begin {2}
   setModal( true );
   // set to true as default to ensure that cancelling using the upper right corner X doesn't result in
        a spurious entry
   setIsDiscarded( true );
   setLocationRelativeTo( getOwner() );

   // user code end
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 19:10:38)
 * @return boolean
 */
public boolean isIsDiscarded() {
   return isDiscarded;
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
/* try {
      DisplayGroup aDisplayGroup;
      aDisplayGroup = new DisplayGroup();
      aDisplayGroup.setModal(true);
      aDisplayGroup.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
```

```
            };
474         });
            aDisplayGroup.setVisible(true);
476      } catch (Throwable exception) {
            System.err.println("Exception occurred in main() of javax.swing.JDialog");
478         exception.printStackTrace(System.out);
         }
480   */
      }
482   /**
       * Insert the method's description here.
484    * Creation date: (26-07-00 17:28:18)
       * @param newGroupDirectory clientapp.GroupDirectory
486    */
      public void setGroupDirectory(clientapp.GroupDirectory newGroupDirectory) {
488      groupDirectory = newGroupDirectory;
      }
490   /**
       * Insert the method's description here.
492    * Creation date: (26-07-00 19:10:38)
       * @param newIsDiscarded boolean
494    */
      public void setIsDiscarded(boolean newIsDiscarded) {
496      isDiscarded = newIsDiscarded;
      }
498   /**
       * Insert the method's description here.
500    * Creation date: (03-07-00 20:56:36)
       * @param newWorkGroup peerviewmisc.WorkGroup
502    */
      public void setWorkGroup(WorkGroup newWorkGroup)
504   {
         workGroup = newWorkGroup;
506
         getGroupNameField().setText( workGroup.getGroupName() );
508      getDescriptionField().setText( workGroup.getDescription() );
      }
510   /**
       * Insert the method's description here.
512    * Creation date: (27-07-00 00:02:29)
       */
514   public abstract void submitAndClose();
      }
```

Listing C.40: DocPackage.java

```
      package peerviewmisc;
 2
      /**
 4     * This class implements a container which represents a node in the client panorama by storing the name
           of its associated
       * document and the size, position and locking status of the node.
 6     * Creation date: (11-10-00 12:03:22)
       * @author:
 8     */
      public class DocPackage implements java.io.Serializable {
10       private java.lang.String docNameAndPath = null;
         private java.awt.Dimension size = null;
12       private java.awt.Dimension position = null;
         private java.lang.String NodeLocked = null;
14    /**
       * DocPackage constructor comment.
16     */
      public DocPackage() {
18       super();
      }
20    /**
```

```
     * Insert the method's description here.
22   * Creation date: (11-10-00 12:12:19)
     * @param docNameAndPathArg java.lang.String
24   * @param size java.awt.Dimension
     * @param position java.awt.Dimension
26   */
    public DocPackage(String docNameAndPathArg, java.awt.Dimension sizeArg, java.awt.Dimension positionArg )
28  {
       setDocNameAndPath( docNameAndPathArg );
30     setSize( sizeArg );
       setPosition( positionArg );
32  }
    /**
34   * Insert the method's description here.
     * Creation date: (11-10-00 12:03:36)
36   * @return java.lang.String
     */
38  public java.lang.String getDocNameAndPath() {
       return docNameAndPath;
40  }
    /**
42   * Insert the method's description here.
     * Creation date: (16-10-00 09:51:21)
44   * @return java.lang.String
     */
46  public java.lang.String getNodeLocked() {
       return NodeLocked;
48  }
    /**
50   * Insert the method's description here.
     * Creation date: (11-10-00 12:04:44)
52   * @return java.awt.Dimension
     */
54  public java.awt.Dimension getPosition() {
       return position;
56  }
    /**
58   * Insert the method's description here.
     * Creation date: (11-10-00 12:03:54)
60   * @return java.awt.Dimension
     */
62  public java.awt.Dimension getSize() {
       return size;
64  }
    /**
66   * Insert the method's description here.
     * Creation date: (11-10-00 12:03:36)
68   * @param newDocNameAndPath java.lang.String
     */
70  public void setDocNameAndPath(java.lang.String newDocNameAndPath) {
       docNameAndPath = newDocNameAndPath;
72  }
    /**
74   * Insert the method's description here.
     * Creation date: (16-10-00 09:51:21)
76   * @param newNodeLocked java.lang.String
     */
78  public void setNodeLocked(java.lang.String newNodeLocked) {
       NodeLocked = newNodeLocked;
80  }
    /**
82   * Insert the method's description here.
     * Creation date: (11-10-00 12:04:44)
84   * @param newPosition java.awt.Dimension
     */
86  public void setPosition(java.awt.Dimension newPosition) {
```

```
      position = newPosition;
   }
   /**
    * Insert the method's description here.
    * Creation date: (11-10-00 12:03:54)
    * @param newSize java.awt.Dimension
    */
   public void setSize(java.awt.Dimension newSize) {
      size = newSize;
   }
   }
```

Listing C.41: EditGroup.java

```
   package peerviewmisc;

   /**
    * The edit group dialog box.
    * Creation date: (29-06-00 16:08:16)
    * @author:
    */
   public class EditGroup extends DisplayGroup
   {

   /**
    * EditGroup constructor comment.
    */
   public EditGroup() {
      super();
      initialize();
   }
   /**
    * EditGroup constructor comment.
    * @param owner java.awt.Dialog
    */
   public EditGroup(java.awt.Dialog owner) {
      super(owner);
   }
   /**
    * EditGroup constructor comment.
    * @param owner java.awt.Dialog
    * @param title java.lang.String
    */
   public EditGroup(java.awt.Dialog owner, String title) {
      super(owner, title);
   }
   /**
    * EditGroup constructor comment.
    * @param owner java.awt.Dialog
    * @param title java.lang.String
    * @param modal boolean
    */
   public EditGroup(java.awt.Dialog owner, String title, boolean modal) {
      super(owner, title, modal);
   }
   /**
    * EditGroup constructor comment.
    * @param owner java.awt.Dialog
    * @param modal boolean
    */
   public EditGroup(java.awt.Dialog owner, boolean modal) {
      super(owner, modal);
   }
   /**
    * EditGroup constructor comment.
    * @param owner java.awt.Frame
    */
```

```
54  public EditGroup(java.awt.Frame owner) {
       super(owner);
56  }
    /**
58   * EditGroup constructor comment.
     * @param owner java.awt.Frame
60   * @param title java.lang.String
     */
62  public EditGroup(java.awt.Frame owner, String title) {
       super(owner, title);
64  }
    /**
66   * EditGroup constructor comment.
     * @param owner java.awt.Frame
68   * @param title java.lang.String
     * @param modal boolean
70   */
    public EditGroup(java.awt.Frame owner, String title, boolean modal) {
72     super(owner, title, modal);
    }
74  /**
     * EditGroup constructor comment.
76   * @param owner java.awt.Frame
     * @param modal boolean
78   */
    public EditGroup(java.awt.Frame owner, boolean modal) {
80     super(owner, modal);
    }
82  /**
     * Called whenever the part throws an exception.
84   * @param exception java.lang.Throwable
     */
86  private void handleException(java.lang.Throwable exception) {

88     /* Uncomment the following lines to print uncaught exceptions to stdout */
       // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
90     // exception.printStackTrace(System.out);
    }
92  /**
     * Initialize the class.
94   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
96  private void initialize() {
       try {
98        // user code begin {1}
          // user code end
100       setName("EditGroup");
       } catch (java.lang.Throwable ivjExc) {
102       handleException(ivjExc);
       }
104    // user code begin {2}
       setTitle( "Edit␣group" );
106    editable( true );
       getOKButton().setText( "Submit" );
108    getOKButton().setMnemonic( 's' );
       getCancelButton().setText( "Discard" );
110    getCancelButton().setMnemonic( 'd' );
       // user code end
112 }
    /**
114  * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
116  */
    public static void main(java.lang.String[] args) {
118    try {
          EditGroup aEditGroup;
```

```
      aEditGroup = new EditGroup();
      aEditGroup.setModal(true);
      aEditGroup.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aEditGroup.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣serverapp.EditGroup");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 00:15:52)
 */
public void submitAndClose()
{
   String groupName = ivjGroupNameField.getText();
   String description = ivjDescriptionField.getText();
   ErrorBox errorBox = new ErrorBox();

   if ( groupName.length() == 0 )
   {
      errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
      errorBox.show();
      return;
   }
   getWorkGroup().setDescription( ivjDescriptionField.getText() );
   getWorkGroup().setGroupName( ivjGroupNameField.getText() );
   setIsDiscarded( false );
   dispose();
}
}
```

Listing C.42: ErrorBox.java

```
package peerviewmisc;

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import javax.swing.event.*;

/**
 * The dialog box displayed when the user is notified of an error.
 * Creation date: (07-06-00 00:37:01)
 * @author:
 */
public class ErrorBox extends JDialog {

   public class CloseButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         dispose();
      }
   }
   private JPanel ivjJDialogContentPane = null;
   private JPanel ivjButtonPanel = null;
   private FlowLayout ivjButtonPanelFlowLayout = null;
   private JLabel ivjIconLabel = null;
   private JPanel ivjMainPanel = null;
```

```
30     private JTextPane ivjMessagePanel = null;
       private JButton ivjCloseButton = null;
32  /**
     * ErrorBox constructor comment.
34   */
    public ErrorBox() {
36     super();
       initialize();
38  }
    /**
40   * ErrorBox constructor comment.
     * @param owner java.awt.Dialog
42   */
    public ErrorBox(java.awt.Dialog owner) {
44     super(owner);
    }
46  /**
     * ErrorBox constructor comment.
48   * @param owner java.awt.Dialog
     * @param title java.lang.String
50   */
    public ErrorBox(java.awt.Dialog owner, String title) {
52     super(owner, title);
    }
54  /**
     * ErrorBox constructor comment.
56   * @param owner java.awt.Dialog
     * @param title java.lang.String
58   * @param modal boolean
     */
60  public ErrorBox(java.awt.Dialog owner, String title, boolean modal) {
       super(owner, title, modal);
62  }
    /**
64   * ErrorBox constructor comment.
     * @param owner java.awt.Dialog
66   * @param modal boolean
     */
68  public ErrorBox(java.awt.Dialog owner, boolean modal) {
       super(owner, modal);
70  }
    /**
72   * ErrorBox constructor comment.
     * @param owner java.awt.Frame
74   */
    public ErrorBox(java.awt.Frame owner) {
76     super(owner);
    }
78  /**
     * ErrorBox constructor comment.
80   * @param owner java.awt.Frame
     * @param title java.lang.String
82   */
    public ErrorBox(java.awt.Frame owner, String title) {
84     super(owner, title);
    }
86  /**
     * ErrorBox constructor comment.
88   * @param owner java.awt.Frame
     * @param title java.lang.String
90   * @param modal boolean
     */
92  public ErrorBox(java.awt.Frame owner, String title, boolean modal) {
       super(owner, title, modal);
94  }
    /**
```

```
 * ErrorBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public ErrorBox(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
   if (ivjButtonPanel == null) {
      try {
         ivjButtonPanel = new javax.swing.JPanel();
         ivjButtonPanel.setName("ButtonPanel");
         ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
         getButtonPanel().add(getCloseButton(), getCloseButton().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjButtonPanel;
}
/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getButtonPanelFlowLayout() {
   java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
   try {
      /* Create part */
      ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
      ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   };
   return ivjButtonPanelFlowLayout;
}
/**
 * Return the OK property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCloseButton() {
   if (ivjCloseButton == null) {
      try {
         ivjCloseButton = new javax.swing.JButton();
         ivjCloseButton.setName("CloseButton");
         ivjCloseButton.setMnemonic('C');
         ivjCloseButton.setText("Close");
         // user code begin {1}
         ivjCloseButton.addActionListener( new CloseButtonHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
```

```
162     return ivjCloseButton;
    }
164 /**
     * Return the IconLabel property value.
166  * @return javax.swing.JLabel
     */
168 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JLabel getIconLabel() {
170     if (ivjIconLabel == null) {
          try {
172         ivjIconLabel = new javax.swing.JLabel();
            ivjIconLabel.setName("IconLabel");
174         ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                general/Stop24.gif")));
            ivjIconLabel.setText("");
176         // user code begin {1}
            // user code end
178       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
180         // user code end
            handleException(ivjExc);
182       }
        }
184     return ivjIconLabel;
    }
186 /**
     * Return the JDialogContentPane property value.
188  * @return javax.swing.JPanel
     */
190 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getJDialogContentPane() {
192     if (ivjJDialogContentPane == null) {
          try {
194         ivjJDialogContentPane = new javax.swing.JPanel();
            ivjJDialogContentPane.setName("JDialogContentPane");
196         ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
            getJDialogContentPane().add(getButtonPanel(), "South");
198         getJDialogContentPane().add(getMainPanel(), "Center");
            // user code begin {1}
200         // user code end
          } catch (java.lang.Throwable ivjExc) {
202         // user code begin {2}
            // user code end
204         handleException(ivjExc);
          }
206     }
        return ivjJDialogContentPane;
208 }
    /**
210  * Return the JPanel1 property value.
     * @return javax.swing.JPanel
212  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
214 private javax.swing.JPanel getMainPanel() {
        if (ivjMainPanel == null) {
216       try {
            ivjMainPanel = new javax.swing.JPanel();
218         ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());
220
            java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
222         constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
            constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
224         constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getIconLabel(), constraintsIconLabel);
226
```

```
            java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
228         constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
            constraintsMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
230         constraintsMessagePanel.weightx = 1.0;
            constraintsMessagePanel.weighty = 1.0;
232         constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
234         // user code begin {1}
            ivjMainPanel.setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
236         // user code end
         } catch (java.lang.Throwable ivjExc) {
238         // user code begin {2}
            // user code end
240         handleException(ivjExc);
         }
242   }
      return ivjMainPanel;
244 }
    /**
246  * Return the JTextPane1 property value.
     * @return javax.swing.JTextPane
248  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
250 private javax.swing.JTextPane getMessagePanel() {
      if (ivjMessagePanel == null) {
252      try {
            ivjMessagePanel = new javax.swing.JTextPane();
254         ivjMessagePanel.setName("MessagePanel");
            ivjMessagePanel.setFont(new java.awt.Font("sansserif.bold", 1, 12));
256         ivjMessagePanel.setBackground(java.awt.Color.lightGray);
            ivjMessagePanel.setEditable(false);
258         // user code begin {1}
            ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
260         // user code end
         } catch (java.lang.Throwable ivjExc) {
262         // user code begin {2}
            // user code end
264         handleException(ivjExc);
         }
266   }
      return ivjMessagePanel;
268 }
    /**
270  * Called whenever the part throws an exception.
     * @param exception java.lang.Throwable
272  */
    private void handleException(java.lang.Throwable exception) {
274
      /* Uncomment the following lines to print uncaught exceptions to stdout */
276   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
      // exception.printStackTrace(System.out);
278 }
    /**
280  * Initialize the class.
     */
282 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void initialize() {
284   try {
         // user code begin {1}
286      // user code end
         setName("ErrorBox");
288      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         setSize(401, 191);
290      setModal(true);
         setTitle("Error␣message");
292      setContentPane(getJDialogContentPane());
```

```
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      // user code begin {2}
      // user code end
   }
   /**
    * main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
    */
   public static void main(java.lang.String[] args) {
      try {
         ErrorBox aErrorBox;
         aErrorBox = new ErrorBox();
         aErrorBox.setModal(true);
         aErrorBox.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
            };
         });
         aErrorBox.setText( "ASFADSRFDFDSFADSFDSFADFDASFSASDFSDFASFSDFADDSFASDFSFADFADFADFDSFDSFASDFS" );

         aErrorBox.setVisible(true);

      } catch (Throwable exception) {
         System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
      }
   }
   /**
    * Insert the method's description here.
    * Creation date: (10-07-00 22:52:54)
    * @param newText java.lang.String
    */
   public void setText(String newText)
   {
      getMessagePanel().setText( newText );
      getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel
           ().getFontMetrics( getMessagePanel().getFont() )));
      getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
      pack();
      setLocationRelativeTo( getOwner() );
   }
   }
```

Listing C.43: HelpDialog.java

```
   package peerviewmisc;

   import javax.swing.tree.*;

   /**
    * This class has now been superseded by a JavaHelp window but was originally intended as an
        implementation of the PeerView
    * help system.
    * Creation date: (29-05-00 12:39:46)
    * @author:
    */
   public class HelpDialog extends javax.swing.JDialog {
      private javax.swing.JPanel ivjHelpDialog = null;
      private javax.swing.JPanel ivjJPanel1 = null;
      private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
      private javax.swing.JSplitPane ivjJSplitPane1 = null;
      private javax.swing.JTextPane ivjJTextPane1 = null;
      private javax.swing.JButton ivjOKButton = null;
      IvjEventHandler ivjEventHandler = new IvjEventHandler();
```

```
20  class IvjEventHandler implements java.awt.event.ActionListener {
          public void actionPerformed(java.awt.event.ActionEvent e) {
22           if (e.getSource() == HelpDialog.this.getOKButton())
                connEtoM1(e);
24        };
       };
26     private javax.swing.JTree ivjTopicTree = null;
    /**
28   * HelpDialog constructor comment.
     */
30  public HelpDialog() {
       super();
32     initialize();
    }
34  /**
     * HelpDialog constructor comment.
36   * @param owner java.awt.Dialog
     */
38  public HelpDialog(java.awt.Dialog owner) {
       super(owner);
40  }
    /**
42   * HelpDialog constructor comment.
     * @param owner java.awt.Dialog
44   * @param title java.lang.String
     */
46  public HelpDialog(java.awt.Dialog owner, String title) {
       super(owner, title);
48  }
    /**
50   * HelpDialog constructor comment.
     * @param owner java.awt.Dialog
52   * @param title java.lang.String
     * @param modal boolean
54   */
    public HelpDialog(java.awt.Dialog owner, String title, boolean modal) {
56     super(owner, title, modal);
    }
58  /**
     * HelpDialog constructor comment.
60   * @param owner java.awt.Dialog
     * @param modal boolean
62   */
    public HelpDialog(java.awt.Dialog owner, boolean modal) {
64     super(owner, modal);
    }
66  /**
     * HelpDialog constructor comment.
68   * @param owner java.awt.Frame
     */
70  public HelpDialog(java.awt.Frame owner) {
       super(owner);
72  }
    /**
74   * HelpDialog constructor comment.
     * @param owner java.awt.Frame
76   * @param title java.lang.String
     */
78  public HelpDialog(java.awt.Frame owner, String title) {
       super(owner, title);
80  }
    /**
82   * HelpDialog constructor comment.
     * @param owner java.awt.Frame
84   * @param title java.lang.String
     * @param modal boolean
```

```
 */
public HelpDialog(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public HelpDialog(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * connEtoM1: (OKButton.action.actionPerformed(java.awt.event.ActionEvent) --> HelpDialog.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM1(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Return the HelpDialog property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getHelpDialog() {
   if (ivjHelpDialog == null) {
      try {
         ivjHelpDialog = new javax.swing.JPanel();
         ivjHelpDialog.setName("HelpDialog");
         ivjHelpDialog.setLayout(new java.awt.BorderLayout());
         getHelpDialog().add(getJPanel1(), "South");
         getHelpDialog().add(getJSplitPane1(), "Center");
         // user code begin {1}
         ivjHelpDialog.setBorder(javax.swing.BorderFactory.createMatteBorder(6,6,6,6,ivjHelpDialog.
              getBackground()));
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjHelpDialog;
}
/**
 * Return the JPanel1 property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel1() {
   if (ivjJPanel1 == null) {
      try {
         ivjJPanel1 = new javax.swing.JPanel();
         ivjJPanel1.setName("JPanel1");
         ivjJPanel1.setLayout(getJPanel1FlowLayout());
```

```
            getJPanel1().add(getOKButton(), getOKButton().getName());
152         // user code begin {1}
            // user code end
154      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
156         // user code end
            handleException(ivjExc);
158      }
         }
160      return ivjJPanel1;
      }
162   /**
       * Return the JPanel1FlowLayout property value.
164    * @return java.awt.FlowLayout
       */
166   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private java.awt.FlowLayout getJPanel1FlowLayout() {
168      java.awt.FlowLayout ivjJPanel1FlowLayout = null;
         try {
170         /* Create part */
            ivjJPanel1FlowLayout = new java.awt.FlowLayout();
172         ivjJPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
         } catch (java.lang.Throwable ivjExc) {
174         handleException(ivjExc);
         };
176      return ivjJPanel1FlowLayout;
      }
178   /**
       * Return the JSplitPane1 property value.
180    * @return javax.swing.JSplitPane
       */
182   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JSplitPane getJSplitPane1() {
184      if (ivjJSplitPane1 == null) {
            try {
186            ivjJSplitPane1 = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
               ivjJSplitPane1.setName("JSplitPane1");
188            getJSplitPane1().add(getJTextPane1(), "bottom");
               getJSplitPane1().add(getTopicTree(), "top");
190            // user code begin {1}
               // user code end
192         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
194            // user code end
               handleException(ivjExc);
196         }
         }
198      return ivjJSplitPane1;
      }
200   /**
       * Return the JTextPane1 property value.
202    * @return javax.swing.JTextPane
       */
204   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTextPane getJTextPane1() {
206      if (ivjJTextPane1 == null) {
            try {
208            ivjJTextPane1 = new javax.swing.JTextPane();
               ivjJTextPane1.setName("JTextPane1");
210            // user code begin {1}
               // user code end
212         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
214            // user code end
               handleException(ivjExc);
216         }
```

```
      }
      return ivjJTextPane1;
   }
   /**
    * Return the OKButton property value.
    * @return javax.swing.JButton
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JButton getOKButton() {
      if (ivjOKButton == null) {
         try {
            ivjOKButton = new javax.swing.JButton();
            ivjOKButton.setName("OKButton");
            ivjOKButton.setText("OK");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjOKButton;
   }
   /**
    * Return the JTree1 property value.
    * @return javax.swing.JTree
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTree getTopicTree() {
      if (ivjTopicTree == null) {
         try {
            ivjTopicTree = new javax.swing.JTree();
            ivjTopicTree.setName("TopicTree");
            // user code begin {1}
            ivjTopicTree.setShowsRootHandles( true );
            ( (DefaultTreeModel) ivjTopicTree.getModel() ).setAsksAllowsChildren( true );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjTopicTree;
   }
   /**
    * Called whenever the part throws an exception.
    * @param exception java.lang.Throwable
    */
   private void handleException(java.lang.Throwable exception) {

      /* Uncomment the following lines to print uncaught exceptions to stdout */
      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
      // exception.printStackTrace(System.out);
   }
   /**
    * Initializes connections
    * @exception java.lang.Exception The exception description.
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private void initConnections() throws java.lang.Exception {
      // user code begin {1}
      // user code end
      getOKButton().addActionListener(ivjEventHandler);
   }
```

```
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
      setName("HelpDialog");
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setSize(426, 240);
      setTitle("Help");
      setContentPane(getHelpDialog());
      initConnections();
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   }
   // user code begin {2}
   setLocationRelativeTo( getOwner() );
   initTree();
   // user code end
}
/**
 * Insert the method's description here.
 * Creation date: (02-09-00 21:20:16)
 */
public void initTree()
{
   javax.swing.tree.DefaultMutableTreeNode rootNode = new javax.swing.tree.DefaultMutableTreeNode( "Not␣
        yet␣fully␣implemented" );
   ( (DefaultTreeModel) getTopicTree().getModel() ).setRoot( rootNode );
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      HelpDialog aHelpDialog;
      aHelpDialog = new HelpDialog();
      aHelpDialog.setModal(true);
      aHelpDialog.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aHelpDialog.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
}
```

Listing C.44: Message.java

```
package peerviewmisc;

 import java.io.Serializable;
 import java.util.Vector;

/**
 * This class implements a representation of the messages that can be displayed in the message area.
 * Creation date: (10-07-00 00:07:52)
 * @author:
 */
public class Message implements Serializable
```

```
{
   private java.lang.String contents = null;
   private java.lang.String author = null;
   private java.lang.String title = null;
   private java.lang.String messageID = null;
   private java.lang.String document = null;
   private java.util.Date creationDate = null;
   private java.lang.String creator = null;

   private final static java.lang.String[] FIELD_NAMES = { "Author", "Title", "Contents", "MessageID", "
        DocumentID", "Created␣on"};
/**
 * Message constructor comment.
 */
public Message() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:09:03)
 * @return java.lang.String
 */
public java.lang.String getAuthor() {
   return author;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:08:45)
 * @return java.lang.String
 */
public java.lang.String getContents() {
   return contents;
}
/**
 * Insert the method's description here.
 * Creation date: (16-07-00 10:40:40)
 * @return java.util.GregorianCalendar
 */
public java.util.Date getCreationDate() {
   return creationDate;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 17:29:31)
 * @return java.lang.String
 */
public java.lang.String getCreator() {
   return creator;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 19:05:13)
 * @return java.util.Vector
 */
public java.util.Vector getDataAsVector()
{
   Vector v = new Vector();

   v.addElement( getAuthor() );
   v.addElement( getTitle() );
   v.addElement( getContents() );
   v.addElement( getMessageID() );
   v.addElement( getDocument() );
   v.addElement( getCreationDate().toString() );

   return v;
```

```
    }
 78 /**
     * Insert the method's description here.
 80  * Creation date: (10-07-00 01:42:07)
     * @return java.lang.String
 82  */
    public java.lang.String getDocument() {
 84    return document;
    }
 86 /**
     * Insert the method's description here.
 88  * Creation date: (19-07-00 18:14:33)
     * @return java.lang.String[]
 90  */
    public final static java.lang.String[] getFIELD_NAMES() {
 92    return FIELD_NAMES;
    }
 94 /**
     * Insert the method's description here.
 96  * Creation date: (10-07-00 00:11:49)
     * @return java.lang.String
 98  */
    public java.lang.String getMessageID()
100 {  // I initially opted for a combination of the toString() method and the document name as unique
         identifier, but
       // the scheme seemed both overly cautious (what is the likelihood of two identically named authors
            submitting
102    // identically entitled messages at the exact same millisecond ?) and aesthetically unpleasing when
            presented
       // to the user.
104    return this.toString() + getDocument();
    }
106 /**
     * Insert the method's description here.
108  * Creation date: (10-07-00 00:11:02)
     * @return java.lang.String
110  */
    public java.lang.String getTitle() {
112    return title;
    }
114 /**
     * Insert the method's description here.
116  * Creation date: (10-07-00 00:09:03)
     * @param newAuthor java.lang.String
118  */
    public void setAuthor(java.lang.String newAuthor) {
120    author = newAuthor;
    }
122 /**
     * Insert the method's description here.
124  * Creation date: (10-07-00 00:08:45)
     * @param newContents java.lang.String
126  */
    public void setContents(java.lang.String newContents) {
128    contents = newContents;
    }
130 /**
     * Insert the method's description here.
132  * Creation date: (16-07-00 10:40:40)
     * @param newCreationDate java.util.GregorianCalendar
134  */
    public void setCreationDate(java.util.Date newCreationDate) {
136    creationDate = newCreationDate;
    }
138 /**
     * Insert the method's description here.
```

```
 * Creation date: (19-07-00 17:29:31)
 * @param newCreator java.lang.String
 */
public void setCreator(java.lang.String newCreator) {
   creator = newCreator;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 01:42:07)
 * @param newDocument java.lang.String
 */
public void setDocument(java.lang.String newDocument) {
   document = newDocument;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:11:02)
 * @param newTitle java.lang.String
 */
public void setTitle(java.lang.String newTitle) {
   title = newTitle;
}
/**
 * Insert the method's description here.
 * Creation date: (11-07-00 12:53:01)
 * @return java.lang.String
 */
public String toString()
{
   return title + ",␣by␣" + author + "␣on␣" + creationDate.toString();
}
}
```

Listing C.45: MessageBox.java

```
package peerviewmisc;

import java.awt.event.*;

/**
 * This class implements a box which is used for displaying brief messages.
 * Creation date: (25-07-00 13:50:50)
 * @author:
 */
public class MessageBox extends javax.swing.JDialog
{
   public class CloseButtonHandler implements ActionListener
   {
      public void actionPerformed( ActionEvent ae )
      {
         dispose();
      }
   }

   private javax.swing.JPanel ivjButtonPanel = null;
   private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
   private javax.swing.JButton ivjCloseButton = null;
   private javax.swing.JLabel ivjIconLabel = null;
   private javax.swing.JPanel ivjJDialogContentPane = null;
   private javax.swing.JPanel ivjJDialogContentPane1 = null;
   private javax.swing.JPanel ivjMainPanel = null;
   private javax.swing.JTextPane ivjMessagePanel = null;
/**
 * MessageBox constructor comment.
 */
public MessageBox() {
   super();
```

```
      initialize();
34  }
    /**
36   * MessageBox constructor comment.
     * @param owner java.awt.Dialog
38   */
    public MessageBox(java.awt.Dialog owner) {
40     super(owner);
    }
42  /**
     * MessageBox constructor comment.
44   * @param owner java.awt.Dialog
     * @param title java.lang.String
46   */
    public MessageBox(java.awt.Dialog owner, String title) {
48     super(owner, title);
    }
50  /**
     * MessageBox constructor comment.
52   * @param owner java.awt.Dialog
     * @param title java.lang.String
54   * @param modal boolean
     */
56  public MessageBox(java.awt.Dialog owner, String title, boolean modal) {
       super(owner, title, modal);
58  }
    /**
60   * MessageBox constructor comment.
     * @param owner java.awt.Dialog
62   * @param modal boolean
     */
64  public MessageBox(java.awt.Dialog owner, boolean modal) {
       super(owner, modal);
66  }
    /**
68   * MessageBox constructor comment.
     * @param owner java.awt.Frame
70   */
    public MessageBox(java.awt.Frame owner) {
72     super(owner);
    }
74  /**
     * MessageBox constructor comment.
76   * @param owner java.awt.Frame
     * @param title java.lang.String
78   */
    public MessageBox(java.awt.Frame owner, String title) {
80     super(owner, title);
    }
82  /**
     * MessageBox constructor comment.
84   * @param owner java.awt.Frame
     * @param title java.lang.String
86   * @param modal boolean
     */
88  public MessageBox(java.awt.Frame owner, String title, boolean modal) {
       super(owner, title, modal);
90  }
    /**
92   * MessageBox constructor comment.
     * @param owner java.awt.Frame
94   * @param modal boolean
     */
96  public MessageBox(java.awt.Frame owner, boolean modal) {
       super(owner, modal);
98  }
```

```
/**
 * Return the ButtonPanel property value.
 * @return javax.swing.JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getButtonPanel() {
   if (ivjButtonPanel == null) {
      try {
         ivjButtonPanel = new javax.swing.JPanel();
         ivjButtonPanel.setName("ButtonPanel");
         ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
         getButtonPanel().add(getCloseButton(), getCloseButton().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjButtonPanel;
}
/**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getButtonPanelFlowLayout() {
   java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
   try {
      /* Create part */
      ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
      ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   };
   return ivjButtonPanelFlowLayout;
}
/**
 * Return the CloseButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCloseButton() {
   if (ivjCloseButton == null) {
      try {
         ivjCloseButton = new javax.swing.JButton();
         ivjCloseButton.setName("CloseButton");
         ivjCloseButton.setMnemonic('C');
         ivjCloseButton.setText("Close");
         // user code begin {1}
         ivjCloseButton.addActionListener( new CloseButtonHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjCloseButton;
}
/**
 * Return the IconLabel property value.
 * @return javax.swing.JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
    private javax.swing.JLabel getIconLabel() {
166    if (ivjIconLabel == null) {
       try {
168       ivjIconLabel = new javax.swing.JLabel();
          ivjIconLabel.setName("IconLabel");
170       ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
              general/Information24.gif")));
          ivjIconLabel.setText("");
172       // user code begin {1}
          // user code end
174    } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
176       // user code end
          handleException(ivjExc);
178    }
       }
180    return ivjIconLabel;
    }
182 /**
     * Return the JDialogContentPane property value.
184  * @return javax.swing.JPanel
     */
186 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getJDialogContentPane() {
188    if (ivjJDialogContentPane == null) {
       try {
190       ivjJDialogContentPane = new javax.swing.JPanel();
          ivjJDialogContentPane.setName("JDialogContentPane");
192       ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
          getJDialogContentPane().add(getJDialogContentPane1(), "Center");
194       // user code begin {1}
          // user code end
196    } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
198       // user code end
          handleException(ivjExc);
200    }
       }
202    return ivjJDialogContentPane;
    }
204 /**
     * Return the JDialogContentPane1 property value.
206  * @return javax.swing.JPanel
     */
208 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JPanel getJDialogContentPane1() {
210    if (ivjJDialogContentPane1 == null) {
       try {
212       ivjJDialogContentPane1 = new javax.swing.JPanel();
          ivjJDialogContentPane1.setName("JDialogContentPane1");
214       ivjJDialogContentPane1.setLayout(new java.awt.BorderLayout());
          getJDialogContentPane1().add(getButtonPanel(), "South");
216       getJDialogContentPane1().add(getMainPanel(), "Center");
          // user code begin {1}
218       // user code end
       } catch (java.lang.Throwable ivjExc) {
220       // user code begin {2}
          // user code end
222       handleException(ivjExc);
       }
224    }
       return ivjJDialogContentPane1;
226 }
    /**
228  * Return the MainPanel property value.
     * @return javax.swing.JPanel
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getMainPanel() {
   if (ivjMainPanel == null) {
      try {
         ivjMainPanel = new javax.swing.JPanel();
         ivjMainPanel.setName("MainPanel");
         ivjMainPanel.setLayout(new java.awt.GridBagLayout());

         java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
         constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
         constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
         constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getIconLabel(), constraintsIconLabel);

         java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
         constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
         constraintsMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
         constraintsMessagePanel.weightx = 1.0;
         constraintsMessagePanel.weighty = 1.0;
         constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
         getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
         // user code begin {1}
         ivjMainPanel.setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjMainPanel;
}
/**
 * Return the MessagePanel property value.
 * @return javax.swing.JTextPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextPane getMessagePanel() {
   if (ivjMessagePanel == null) {
      try {
         ivjMessagePanel = new javax.swing.JTextPane();
         ivjMessagePanel.setName("MessagePanel");
         ivjMessagePanel.setFont(new java.awt.Font("sansserif.bold", 1, 12));
         ivjMessagePanel.setBackground(java.awt.Color.lightGray);
         ivjMessagePanel.setEditable(false);
         // user code begin {1}
         ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjMessagePanel;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
```

```
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
      setName("MessageBox");
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setSize(390, 245);
      setModal(true);
      setTitle("Message␣box");
      setContentPane(getJDialogContentPane());
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   }
   // user code begin {2}
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      MessageBox aMessageBox;
      aMessageBox = new MessageBox();
      aMessageBox.setModal(true);
      aMessageBox.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aMessageBox.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (25-07-00 14:22:09)
 * @param newText java.lang.String
 */
public void setText(String newText)
{
   getMessagePanel().setText( newText );
   getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel
        ().getFontMetrics( getMessagePanel().getFont() )));
   getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
   getMessagePanel().setAlignmentY( javax.swing.JComponent.CENTER_ALIGNMENT );
   pack();
   setLocationRelativeTo( getOwner() );
}
}
```

Listing C.46: NewGroup.java

```
package peerviewmisc;

import java.awt.event.*;

/**
 * The dialog box for specifying and submitting a new group.
 * Creation date: (29-06-00 15:52:46)
```

```
 * @author:
 */
public class NewGroup extends DisplayGroup
{

/**
 * NewGroup constructor comment.
 */
public NewGroup() {
   super();
   initialize();
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Dialog
 */
public NewGroup(java.awt.Dialog owner) {
   super(owner);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public NewGroup(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public NewGroup(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public NewGroup(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Frame
 */
public NewGroup(java.awt.Frame owner) {
   super(owner);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public NewGroup(java.awt.Frame owner, String title) {
   super(owner, title);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public NewGroup(java.awt.Frame owner, String title, boolean modal) {
```

```
   super(owner, title, modal);
}
/**
 * NewGroup constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public NewGroup(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(java.lang.Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout */
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   try {
      // user code begin {1}
      // user code end
      setName("NewGroup");
      setModal(true);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   }
   // user code begin {2}
   setTitle( "New␣group" );
   editable( true );
   getOKButton().setText( "Submit" );
   getOKButton().setMnemonic( 's' );
   getCancelButton().setText( "Discard" );
   getCancelButton().setMnemonic( 'd' );

   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      NewGroup aNewGroup;
      aNewGroup = new NewGroup();
      aNewGroup.setModal(true);
      aNewGroup.addWindowListener(new java.awt.event.WindowAdapter() {
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         };
      });
      aNewGroup.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception␣occurred␣in␣main()␣of␣serverapp.NewGroup");
      exception.printStackTrace(System.out);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 00:15:52)
```

```
 */
public void submitAndClose()
{
   String groupName = ivjGroupNameField.getText();
   String description = ivjDescriptionField.getText();
   ErrorBox errorBox = new ErrorBox();

   if ( groupName.length() == 0 )
   {
      errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
      errorBox.show();
      return;
   }
   if ( getGroupDirectory().groupExists( groupName, description )== true )
   {
      errorBox.setText( SharedConstants.getGROUP_EXISTS_MESSAGE() );
      errorBox.show();
      return;
   }
   getWorkGroup().setDescription( ivjDescriptionField.getText() );
   getWorkGroup().setGroupName( ivjGroupNameField.getText() );
   setIsDiscarded( false );
   dispose();
}
}
```

Listing C.47: Pair.java

```
package peerviewmisc;

/**
 * A container primitive mapping a key to a value.
 * Creation date: (20-09-00 02:31:55)
 * @author:
 */
public class Pair implements java.io.Serializable {
   private java.lang.String key = null;
   private java.lang.String value = null;
/**
 * Pair constructor comment.
 */
public Pair() {
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 02:36:16)
 * @param keyArg java.lang.String
 * @param valueArg java.lang.String
 */
public Pair(String keyArg, String valueArg)
{
   key = keyArg;
   value = valueArg;
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 02:32:13)
 * @return java.lang.String
 */
public java.lang.String getKey() {
   return key;
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 02:34:30)
 * @return java.lang.String
```

```
 */
public java.lang.String getValue() {
   return value;
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 02:32:13)
 * @param newKey java.lang.String
 */
public void setKey(java.lang.String newKey) {
   key = newKey;
}
/**
 * Insert the method's description here.
 * Creation date: (20-09-00 02:34:30)
 * @param newValue java.lang.String
 */
public void setValue(java.lang.String newValue) {
   value = newValue;
}
}
```

Listing C.48: SharedConstants.java

```
package peerviewmisc;

 import javax.swing.border.*;
 import javax.swing.BorderFactory.*;
 import com.sun.media.jsdt.*;
 import java.util.Hashtable;

 /**
 * An entity object containing constants shared by the client and server applications.
 * Creation date: (30-06-00 21:39:29)
 * @author:
 */
public class SharedConstants {
   protected final static int BORDER_WIDTH = 6;
   protected final static java.lang.String GROUP_MANAGEMENT_SESSION_NAME = "group␣management";
   protected final static java.lang.String DEFAULT_SERVER_NAME = "localhost";
   protected final static int DEFAULT_SERVER_PORT = 4461;
   protected final static java.lang.String CONNECTION_TYPE = "Connection␣type";
   protected final static java.lang.String REQUEST_GROUP_DIRECTORY = "request␣group␣directory";

   protected final static java.lang.String SERVER_NAME = "Server␣name";
   protected final static java.lang.String SERVER_PORT = "Server␣port";
   protected final static java.lang.String DEFAULT_CONNECTION_TYPE = "socket";
   private final static java.lang.String REQUEST_ADD_GROUP = "Request␣add␣group";
   private final static java.lang.String GROUP_DIRECTORY = "Group␣directory";
   private final static java.lang.String NEW_DOCUMENT = "New␣document";
   private final static java.lang.String GROUP_CHANNEL_NAME = "GroupChannel";
   private final static java.lang.String WORKGROUP_CHANNEL_NAME = "Workgroup␣channel";
   private final static java.lang.String CLIENT_ADDED_DOCUMENTS = "Client␣added␣documents";
   private final static java.lang.String REQUEST_TOPIC_TREE = "Request␣topic␣tree";
   private final static java.lang.String TOPIC_TREE = "Topic␣tree";
   private final static java.lang.String REQUEST_MESSAGE = "Request␣message";
   private final static java.lang.String MESSAGE = "Message";
   private final static java.lang.String DISCUSSION_UPDATE_FROM_CLIENT = "Discussion␣update␣from␣client"
        ;
   private final static java.lang.String CLIENT_ADDED_MESSAGE = "Client␣added␣message";
   private final static java.lang.String TOPIC_TREE_UPDATE = "Topic␣tree␣update";
   private final static java.lang.String DELETE_MESSAGE = "Delete␣message";
   private final static java.lang.String CLIENT_REMOVED_LOCAL_DOCUMENTS = "Client␣removed␣local␣
        documents";
   public final static java.lang.String CLIENT_REMOVED_DOCUMENT = "Client␣removed␣document";
   protected final static java.lang.String DEFAULT_CLIENT_NAME = "defaultName";
   private final static java.lang.String WORKGROUP_UPDATE_FROM_CLIENT = "Workgroup␣update␣from␣client";
```

```
    private final static java.lang.String WORKGROUP_UPDATE = "Workgroup␣update";
    private final static java.lang.String CLIENT_LEFT_GROUP = "Client␣left␣group";
    public final static java.lang.String REMOVE_DOCUMENT = "Document␣was␣removed";
    public final static java.lang.String GROUP_NAME_MISSING_MESSAGE = "You␣must␣specify␣a␣name␣before␣the
        ␣group␣can␣be␣created.";
    private final static java.lang.String GROUP_EXISTS_MESSAGE = "A␣group␣by␣this␣name␣and␣description␣
        already␣exists.";
    private final static java.lang.String MAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE = "The␣number␣of␣
        maximum␣participants␣must␣be␣an␣integer␣value␣greater␣than␣0.";
    private final static java.lang.String REQUEST_REMOVE_GROUP = "Remove␣workgroup␣from␣directory";

      private final static java.lang.String SOCKET = "socket";
    private final static java.lang.String HTTP = "http";
    private final static java.lang.String[] CONNECTION_TYPES = {getSOCKET(),getHTTP()};

    private final static java.lang.String[][] JSDT_EXCEPTION_MESSAGE = {
      { new AlreadyBoundException().getClass().toString(), "An␣attempt␣to␣register␣a␣group␣or␣group␣
          member␣conflicted␣with␣one␣or␣more␣existing␣entries" },
      { new ConnectionException().getClass().toString(), "A␣connection␣error␣occurred␣while␣attempting␣
          to␣connect␣to␣another␣member␣or␣the␣server" },
      { new InvalidClientException().getClass().toString(), "An␣attempt␣to␣access␣or␣reference␣a␣group␣
          member␣resulted␣in␣failure"},
      { new InvalidURLException().getClass().toString(), "Tried␣to␣use␣a␣malformed␣URL␣for␣network␣
          access" },
      { new NameInUseException().getClass().toString(), "There␣was␣a␣naming␣conflict␣between␣two␣or␣more
          ␣Peerview␣entities" },
      { new NoRegistryException().getClass().toString(), "Could␣not␣connect␣to␣the␣registry" },
      { new NoSuchChannelException().getClass().toString(), "An␣attempt␣was␣made␣to␣access␣a␣non-
          existent␣group␣communications␣channel" },
      { new NoSuchClientException().getClass().toString(), "An␣attempt␣was␣made␣to␣reference␣or␣
          communicate␣with␣a␣non-existent␣group␣member" },
      { new NoSuchConsumerException().getClass().toString(), "An␣attempt␣was␣made␣to␣reference␣or␣
          communicate␣with␣a␣non-existent␣group" },
      { new NoSuchHostException().getClass().toString(), "Could␣not␣connect␣to␣the␣specified␣server" },
      { new NoSuchSessionException().getClass().toString(), "An␣attempt␣was␣made␣to␣reference␣or␣
          communicate␣with␣a␣non-existent␣group" },
      { new NotBoundException().getClass().toString(), "␣An␣attempt␣was␣made␣to␣reference␣or␣communicate
          ␣with␣one␣or␣more␣unregistered␣groups␣or␣group␣members" },
      { new PermissionDeniedException().getClass().toString(), "Attempted␣an␣impermissible␣operation" },
      { new PortInUseException().getClass().toString(), "Tried␣to␣communicate␣using␣a␣server␣port␣that␣
          was␣already␣in␣use" },
      { new RegistryExistsException().getClass().toString(), "An␣attempt␣to␣create␣a␣new␣registry␣
          conflicted␣with␣an␣existing␣one" },
      { new TimedOutException().getClass().toString(), "An␣operation␣timed␣out␣before␣it␣could␣be␣
          completed" },
    };
    private final static java.util.Hashtable JSDT_EXCEPTION_MESSAGES_TABLE = new Hashtable();
    private final static int NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS = 50;
    private final static java.lang.String COMPRESSION_LEVEL = "Compression␣level";
    private final static int DEFAULT_COMPRESSION_LEVEL = java.util.zip.Deflater.BEST_COMPRESSION;
    private final static java.lang.String DISCUSSIONS_FILE_NAME = "discuss.dat";
    private final static java.lang.String SOCKET_ARGUMENT = "-socket";
    private final static java.lang.String HTTP_ARGUMENT = "-http";

    private final static java.lang.String REGISTRY_PORT = "Registry␣port";
    private final static java.lang.String DEFAULT_REGISTRY_PORT = "4561";
    private final static java.lang.String WORKGROUP_INCONSISTENT = "Workgroup␣inconsistent";
    private final static java.lang.String CLIENT_JOINED_GROUP = "Client␣joined␣group";
    private final static java.lang.String NEW_DOCUMENTS = "New␣documents";
    private final static java.lang.String REMOVE_DOCUMENTS = "Remove␣documents";
    private final static java.lang.String CLIENT_LEFT = "Client␣left";
    private final static java.lang.String COULD_NOT_EXIT_PROPERLY = "The␣application␣could␣not␣be␣
        properly␣terminated.␣There␣may␣be␣orphaned␣processes␣running␣on␣your␣system.";
/**
 * SharedConstants constructor comment.
 */
```

```
    public SharedConstants() {
 92    super();
    }
 94 /**
     * Similar to computeDisplayRectangle, but with a Dimension type return value.
 96  * Creation date: (10-08-00 06:48:34)
     * @return java.awt.Dimension
 98  * @param string java.lang.String
     * @param fontMetrics java.awt.FontMetrics
100  * @param numRows int
     */
102 public static java.awt.Dimension computeDisplayDimension(String string, java.awt.FontMetrics fontMetrics
        )
    {
104    java.awt.Dimension size = new java.awt.Dimension();

106    int numRows = string.length() / SharedConstants.getNOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS
           ();
       size.width = fontMetrics.stringWidth( string )/ (numRows + 1) ;
108    size.height = fontMetrics.getHeight() * (numRows*2 + 3);

110    return size;
    }
112 /**
     * Insert the method's description here.
114  * Creation date: (29-06-00 11:34:48)
     * @return javax.swing.border.Border
116  */
    public static javax.swing.border.Border createDialogBorder(java.awt.Color colour)
118 {
       Border matteBorder, etchedBorder, compoundBorder, returnBorder;
120
       matteBorder = javax.swing.BorderFactory.createMatteBorder( getBORDER_WIDTH(), getBORDER_WIDTH(),
           getBORDER_WIDTH(), getBORDER_WIDTH(), colour );
122    etchedBorder = javax.swing.BorderFactory.createEtchedBorder();
       compoundBorder = javax.swing.BorderFactory.createCompoundBorder( matteBorder, etchedBorder );
124    returnBorder = javax.swing.BorderFactory.createCompoundBorder( compoundBorder, matteBorder );

126    return returnBorder;
    }
128 /**
     * Constructs a representative, unique ID for a document based on the name of the document and that of
         its sender, i.e. a client.
130  * Creation date: (09-07-00 15:17:39)
     * @return java.lang.String
132  * @param senderName java.lang.String
     * @param documentName java.lang.String
134  */
    public static String createFullyQualifiedDocumentName(String senderName, String documentName)
136 {
       return ("Sender:" + senderName + ".Document:" + documentName);
138 }
    /**
140  * Insert the method's description here.
     * Creation date: (30-06-00 21:42:27)
142  * @return int
     */
144 public final static int getBORDER_WIDTH() {
       return BORDER_WIDTH;
146 }
    /**
148  * Insert the method's description here.
     * Creation date: (09-07-00 14:36:29)
150  * @return java.lang.String
     */
152 public final static java.lang.String getCLIENT_ADDED_DOCUMENTS() {
```

```
        return CLIENT_ADDED_DOCUMENTS;
154  }
     /**
156   * Insert the method's description here.
      * Creation date: (11-07-00 21:13:58)
158   * @return java.lang.String
      */
160  public final static java.lang.String getCLIENT_ADDED_MESSAGE() {
        return CLIENT_ADDED_MESSAGE;
162  }
     /**
164   * Insert the method's description here.
      * Creation date: (19-09-00 16:23:30)
166   * @return java.lang.String
      */
168  public final static java.lang.String getCLIENT_JOINED_GROUP() {
        return CLIENT_JOINED_GROUP;
170  }
     /**
172   * Insert the method's description here.
      * Creation date: (01-11-00 01:20:01)
174   * @return java.lang.String
      */
176  public final static java.lang.String getCLIENT_LEFT() {
        return CLIENT_LEFT;
178  }
     /**
180   * Insert the method's description here.
      * Creation date: (26-07-00 01:41:43)
182   * @return java.lang.String
      */
184  public final static java.lang.String getCLIENT_LEFT_GROUP() {
        return CLIENT_LEFT_GROUP;
186  }
     /**
188   * Insert the method's description here.
      * Creation date: (15-07-00 16:26:04)
190   * @return java.lang.String
      */
192  public final static java.lang.String getCLIENT_REMOVED_DOCUMENT() {
        return CLIENT_REMOVED_DOCUMENT;
194  }
     /**
196   * Insert the method's description here.
      * Creation date: (15-07-00 15:33:51)
198   * @return java.lang.String
      */
200  public final static java.lang.String getCLIENT_REMOVED_LOCAL_DOCUMENTS() {
        return CLIENT_REMOVED_LOCAL_DOCUMENTS;
202  }
     /**
204   * Insert the method's description here.
      * Creation date: (19-08-00 12:26:31)
206   * @return java.lang.String
      */
208  public final static java.lang.String getCOMPRESSION_LEVEL() {
        return COMPRESSION_LEVEL;
210  }
     /**
212   * Insert the method's description here.
      * Creation date: (03-07-00 09:29:42)
214   * @return java.lang.String
      */
216  public final static java.lang.String getCONNECTION_TYPE() {
        return CONNECTION_TYPE;
218  }
```

```
    /**
220  * Insert the method's description here.
     * Creation date: (05-08-00 20:51:35)
222  * @return java.lang.String[]
     */
224 public final static java.lang.String[] getCONNECTION_TYPES() {
       return CONNECTION_TYPES;
226 }
    /**
228  * Insert the method's description here.
     * Creation date: (11/23/00 1:56:30 PM)
230  * @return java.lang.String
     */
232 public final static java.lang.String getCOULD_NOT_EXIT_PROPERLY() {
       return COULD_NOT_EXIT_PROPERLY;
234 }
    /**
236  * Insert the method's description here.
     * Creation date: (16-07-00 11:32:37)
238  * @return java.lang.String
     */
240 public final static java.lang.String getDEFAULT_CLIENT_NAME() {
       return DEFAULT_CLIENT_NAME;
242 }
    /**
244  * Insert the method's description here.
     * Creation date: (19-08-00 12:26:58)
246  * @return int
     */
248 public final static int getDEFAULT_COMPRESSION_LEVEL() {
       return DEFAULT_COMPRESSION_LEVEL;
250 }
    /**
252  * Insert the method's description here.
     * Creation date: (03-07-00 16:51:41)
254  * @return java.lang.String
     */
256 public final static java.lang.String getDEFAULT_CONNECTION_TYPE() {
       return DEFAULT_CONNECTION_TYPE;
258 }
    /**
260  * Insert the method's description here.
     * Creation date: (12-09-00 10:17:15)
262  * @return java.lang.String
     */
264 public final static java.lang.String getDEFAULT_REGISTRY_PORT() {
       return DEFAULT_REGISTRY_PORT;
266 }
    /**
268  * Insert the method's description here.
     * Creation date: (03-07-00 09:26:36)
270  * @return java.lang.String
     */
272 public final static java.lang.String getDEFAULT_SERVER_NAME() {
       return DEFAULT_SERVER_NAME;
274 }
    /**
276  * Insert the method's description here.
     * Creation date: (03-07-00 09:28:44)
278  * @return int
     */
280 public final static int getDEFAULT_SERVER_PORT() {
       return DEFAULT_SERVER_PORT;
282 }
    /**
284  * Insert the method's description here.
```

```
     * Creation date: (12-07-00 12:11:27)
286  * @return java.lang.String
     */
288 public final static java.lang.String getDELETE_MESSAGE() {
       return DELETE_MESSAGE;
290 }
    /**
292  * Insert the method's description here.
     * Creation date: (11-07-00 14:32:07)
294  * @return java.lang.String
     */
296 public final static java.lang.String getDISCUSSION_UPDATE_FROM_CLIENT() {
       return DISCUSSION_UPDATE_FROM_CLIENT;
298 }
    /**
300  * Insert the method's description here.
     * Creation date: (24-08-00 08:41:42)
302  * @return java.lang.String
     */
304 public final static java.lang.String getDISCUSSIONS_FILE_NAME() {
       return DISCUSSIONS_FILE_NAME;
306 }
    /**
308  * Insert the method's description here.
     * Creation date: (07-07-00 13:46:31)
310  * @return java.lang.String
     */
312 public final static java.lang.String getGROUP_CHANNEL_NAME() {
       return GROUP_CHANNEL_NAME;
314 }
    /**
316  * Insert the method's description here.
     * Creation date: (04-07-00 17:33:19)
318  * @return java.lang.String
     */
320 public final static java.lang.String getGROUP_DIRECTORY() {
       return GROUP_DIRECTORY;
322 }
    /**
324  * Insert the method's description here.
     * Creation date: (26-07-00 16:40:54)
326  * @return java.lang.String
     */
328 public final static java.lang.String getGROUP_EXISTS_MESSAGE() {
       return GROUP_EXISTS_MESSAGE;
330 }
    /**
332  * Insert the method's description here.
     * Creation date: (30-06-00 22:43:36)
334  * @return java.lang.String
     */
336 public final static java.lang.String getGROUP_MANAGEMENT_SESSION_NAME() {
       return GROUP_MANAGEMENT_SESSION_NAME;
338 }
    /**
340  * Insert the method's description here.
     * Creation date: (26-07-00 16:39:50)
342  * @return java.lang.String
     */
344 public final static java.lang.String getGROUP_NAME_MISSING_MESSAGE() {
       return GROUP_NAME_MISSING_MESSAGE;
346 }
    /**
348  * Insert the method's description here.
     * Creation date: (10-09-00 20:04:03)
350  * @return java.lang.String
```

```
     */
352  public final static java.lang.String getHTTP() {
       return HTTP;
354  }
     /**
356   * Insert the method's description here.
      * Creation date: (10-09-00 20:03:32)
358   * @return java.lang.String
      */
360  public final static java.lang.String getHTTP_ARGUMENT() {
       return HTTP_ARGUMENT;
362  }
     /**
364   * Insert the method's description here.
      * Creation date: (08-08-00 16:57:01)
366   * @return java.lang.String[][]
      */
368  public final static java.lang.String[][] getJSDT_EXCEPTION_MESSAGE() {
       return JSDT_EXCEPTION_MESSAGE;
370  }
     /**
372   * Insert the method's description here.
      * Creation date: (09-08-00 07:59:38)
374   * @return java.util.Hashtable
      */
376  public final static java.util.Hashtable getJSDT_EXCEPTION_MESSAGES_TABLE()
     {
378    if ( JSDT_EXCEPTION_MESSAGES_TABLE.size() == 0 )
       {
380      for ( int i = 0 ; i < JSDT_EXCEPTION_MESSAGE.length; i++ )
         {
382        JSDT_EXCEPTION_MESSAGES_TABLE.put( JSDT_EXCEPTION_MESSAGE[i][0], JSDT_EXCEPTION_MESSAGE[i][1] );
         }
384    }
       return JSDT_EXCEPTION_MESSAGES_TABLE;
386  }
     /**
388   * Insert the method's description here.
      * Creation date: (26-07-00 16:57:40)
390   * @return java.lang.String
      */
392  public final static java.lang.String getMAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE() {
       return MAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE;
394  }
     /**
396   * Insert the method's description here.
      * Creation date: (10-07-00 02:26:24)
398   * @return java.lang.String
      */
400  public final static java.lang.String getMESSAGE() {
       return MESSAGE;
402  }
     /**
404   * Insert the method's description here.
      * Creation date: (06-07-00 23:45:33)
406   * @return java.lang.String
      */
408  public final static java.lang.String getNEW_DOCUMENT() {
       return NEW_DOCUMENT;
410  }
     /**
412   * Insert the method's description here.
      * Creation date: (19-09-00 18:28:18)
414   * @return java.lang.String
      */
416  public final static java.lang.String getNEW_DOCUMENTS() {
```

```
        return NEW_DOCUMENTS;
418  }
     /**
420   * Insert the method's description here.
      * Creation date: (10-08-00 06:56:22)
422   * @return int
      */
424  public final static int getNOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS() {
        return NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS;
426  }
     /**
428   * Insert the method's description here.
      * Creation date: (12-09-00 10:16:20)
430   * @return java.lang.String
      */
432  public final static java.lang.String getREGISTRY_PORT() {
        return REGISTRY_PORT;
434  }
     /**
436   * Insert the method's description here.
      * Creation date: (26-07-00 02:41:39)
438   * @return java.lang.String
      */
440  public final static java.lang.String getREMOVE_DOCUMENT() {
        return REMOVE_DOCUMENT;
442  }
     /**
444   * Insert the method's description here.
      * Creation date: (19-09-00 19:15:50)
446   * @return java.lang.String
      */
448  public final static java.lang.String getREMOVE_DOCUMENTS() {
        return REMOVE_DOCUMENTS;
450  }
     /**
452   * Insert the method's description here.
      * Creation date: (04-07-00 17:33:01)
454   * @return java.lang.String
      */
456  public final static java.lang.String getREQUEST_ADD_GROUP() {
        return REQUEST_ADD_GROUP;
458  }
     /**
460   * Insert the method's description here.
      * Creation date: (03-07-00 12:39:46)
462   * @return java.lang.String
      */
464  public final static java.lang.String getREQUEST_GROUP_DIRECTORY() {
        return REQUEST_GROUP_DIRECTORY;
466  }
     /**
468   * Insert the method's description here.
      * Creation date: (10-07-00 02:16:46)
470   * @return java.lang.String
      */
472  public final static java.lang.String getREQUEST_MESSAGE() {
        return REQUEST_MESSAGE;
474  }
     /**
476   * Insert the method's description here.
      * Creation date: (27-07-00 00:26:45)
478   * @return java.lang.String
      */
480  public final static java.lang.String getREQUEST_REMOVE_GROUP() {
        return REQUEST_REMOVE_GROUP;
482  }
```

```
    /**
484  * Insert the method's description here.
     * Creation date: (09-07-00 21:54:55)
486  * @return java.lang.String
     */
488 public final static java.lang.String getREQUEST_TOPIC_TREE() {
       return REQUEST_TOPIC_TREE;
490 }
    /**
492  * Insert the method's description here.
     * Creation date: (03-07-00 14:30:27)
494  * @return java.lang.String
     */
496 public final static java.lang.String getSERVER_NAME() {
       return SERVER_NAME;
498 }
    /**
500  * Insert the method's description here.
     * Creation date: (03-07-00 14:30:42)
502  * @return java.lang.String
     */
504 public final static java.lang.String getSERVER_PORT() {
       return SERVER_PORT;
506 }
    /**
508  * Insert the method's description here.
     * Creation date: (10-09-00 20:03:50)
510  * @return java.lang.String
     */
512 public final static java.lang.String getSOCKET() {
       return SOCKET;
514 }
    /**
516  * Insert the method's description here.
     * Creation date: (10-09-00 20:03:13)
518  * @return java.lang.String
     */
520 public final static java.lang.String getSOCKET_ARGUMENT() {
       return SOCKET_ARGUMENT;
522 }
    /**
524  * Insert the method's description here.
     * Creation date: (09-07-00 22:29:03)
526  * @return java.lang.String
     */
528 public final static java.lang.String getTOPIC_TREE() {
       return TOPIC_TREE;
530 }
    /**
532  * Insert the method's description here.
     * Creation date: (11-07-00 21:31:23)
534  * @return java.lang.String
     */
536 public final static java.lang.String getTOPIC_TREE_UPDATE() {
       return TOPIC_TREE_UPDATE;
538 }
    /**
540  * Insert the method's description here.
     * Creation date: (08-07-00 17:28:19)
542  * @return java.lang.String
     */
544 public final static java.lang.String getWORKGROUP_CHANNEL_NAME() {
       return WORKGROUP_CHANNEL_NAME;
546 }
    /**
548  * Insert the method's description here.
```

```
     * Creation date: (18-09-00 20:53:55)
550  * @return java.lang.String
     */
552  public final static java.lang.String getWORKGROUP_INCONSISTENT() {
        return WORKGROUP_INCONSISTENT;
554  }
     /**
556  * Insert the method's description here.
     * Creation date: (22-07-00 20:15:39)
558  * @return java.lang.String
     */
560  public final static java.lang.String getWORKGROUP_UPDATE() {
        return WORKGROUP_UPDATE;
562  }
     /**
564  * Insert the method's description here.
     * Creation date: (22-07-00 17:52:54)
566  * @return java.lang.String
     */
568  public final static java.lang.String getWORKGROUP_UPDATE_FROM_CLIENT() {
        return WORKGROUP_UPDATE_FROM_CLIENT;
570  }
     }
```

Listing C.49: WorkGroup.java

```
     package peerviewmisc;
 2
      import java.io.*;
 4    import java.util.Hashtable;
      import com.sun.media.jsdt.*;
 6    import java.util.Vector;
      import java.util.Date;
 8
     /**
10   * This class implements a representation of a PeerView group.
      * Creation date: (01-07-00 09:22:26)
12   * @author:
      */
14   public class WorkGroup implements Serializable
     {
16      private java.lang.String groupName = "";
        private java.lang.String description = "";
18      private com.sun.media.jsdt.URLString sessionURL = null;
        private long totalDataVolume = 0;
20      private java.util.Date creationDate = new Date();
        private java.lang.String creator = "";
22      private final static java.lang.String fieldNames[] = { "Group␣name", "Description",
                                        "#␣of␣documents", "Total␣data␣volume", "Active␣participants",
24                                      "Creation␣date", "Created␣by" };
        private java.util.Hashtable documents = new Hashtable();
26      private java.util.HashSet participants = new java.util.HashSet();
        private int numberOfDocuments = 0;
28   /**
      * WorkGroup constructor comment.
30   */
     public WorkGroup() {
32      super();
     }
34   /**
      * Insert the method's description here.
36   * Creation date: (19-09-00 19:35:37)
      * @param workGroup peerviewmisc.WorkGroup
38   */
     public WorkGroup(WorkGroup workGroup)
40   {
        setCreationDate( workGroup.getCreationDate() );
```

```
42      setCreator( workGroup.getCreator() );
        setDescription( workGroup.getDescription() );
44      setDocuments( (Hashtable) workGroup.getDocuments().clone() );
        setGroupName( workGroup.getGroupName() );
46      setParticipants( (java.util.HashSet) workGroup.getParticipants().clone() );
        setSessionURL( workGroup.getSessionURL() );
48      setTotalDataVolume( workGroup.getTotalDataVolume() );
    }
50  /**
     * Insert the method's description here.
52   * Creation date: (16-07-00 11:01:38)
     * @return java.util.Date
54   */
    public java.util.Date getCreationDate() {
56      return creationDate;
    }
58  /**
     * Insert the method's description here.
60   * Creation date: (17-07-00 17:12:13)
     * @return java.lang.String
62   */
    public java.lang.String getCreator() {
64      return creator;
    }
66  /**
     * Insert the method's description here.
68   * Creation date: (05-07-00 21:32:10)
     * @return java.lang.String
70   */
    public java.lang.String getDescription() {
72      return description;
    }
74  /**
     * Insert the method's description here.
76   * Creation date: (19-09-00 17:40:00)
     * @return java.util.Hashtable
78   */
    public java.util.Hashtable getDocuments() {
80      return documents;
    }
82  /**
     * Insert the method's description here.
84   * Creation date: (17-07-00 19:32:07)
     * @return java.lang.String
86   */
    public final static java.lang.String[] getFieldNames() {
88      return fieldNames;
    }
90  /**
     * Pack the data fields corresponding to the entries in the field label array into a vector and return
         it. This method
92   * primarily intended for use with tabular display (a vector corresponds to a row of table data).
     * Creation date: (17-07-00 20:24:14)
94   * @return java.util.Vector
     */
96  public java.util.Vector getGroupAsVector()
    {
98      Vector v = new Vector();

100     v.addElement( getGroupName() );
        v.addElement( getDescription() );
102     v.addElement( String.valueOf( getNumberOfDocuments() ));
        v.addElement( String.valueOf( getTotalDataVolume() ));
104     v.addElement( String.valueOf( participants.size() ));
        v.addElement( getCreationDate().toString() );
106     v.addElement( getCreator() );
```

```
108     return v;

110

112  }
     /**
114   * Insert the method's description here.
      * Creation date: (05-07-00 21:25:46)
116   * @return java.lang.String
      */
118  public java.lang.String getGroupName() {
        return groupName;
120  }
     /**
122   * Insert the method's description here.
      * Creation date: (19-09-00 18:45:20)
124   * @return int
      */
126  public int getNumberOfDocuments() {
        return numberOfDocuments;
128  }
     /**
130   * Insert the method's description here.
      * Creation date: (19-09-00 17:53:36)
132   * @return java.util.HashSet
      */
134  public java.util.HashSet getParticipants() {
        return participants;
136  }
     /**
138   * Insert the method's description here.
      * Creation date: (07-07-00 20:11:38)
140   * @return com.sun.media.jsdt.URLString
      */
142  public com.sun.media.jsdt.URLString getSessionURL() {
        return sessionURL;
144  }
     /**
146   * Insert the method's description here.
      * Creation date: (08-07-00 20:06:01)
148   * @return long
      */
150  public long getTotalDataVolume() {
        return totalDataVolume;
152  }
     /**
154   * Create and return unique identifier for the JSDT channel created to convey information pertaining to
          workgroup.
      * Creation date: (13-08-00 01:42:08)
156   * @return java.lang.String
      */
158  public String getWorkGroupChannelID()
     {
160     return getWorkGroupID() + SharedConstants.getWORKGROUP_CHANNEL_NAME();
     }
162  /**
      * Same as getWorkGroupChannelID(), only with a specifiable suffix.
164   * Creation date: (13-08-00 01:43:08)
      * @return java.lang.String
166   * @param suffix java.lang.String
      */
168  public String getWorkGroupChannelID(String suffix)
     {
170     return getWorkGroupID() + suffix;
     }
```

```
/**
 * Create and return a unique identifier for the workgroup
 * Creation date: (17-07-00 17:53:57)
 * @return java.lang.String
 */
public String getWorkGroupID()
{
   return getGroupName() + getDescription() + getCreator();
}
/**
 * Insert the method's description here.
 * Creation date: (16-07-00 11:01:38)
 * @param newCreationDate java.util.Date
 */
public void setCreationDate(java.util.Date newCreationDate) {
   creationDate = newCreationDate;
}
/**
 * Insert the method's description here.
 * Creation date: (17-07-00 17:12:13)
 * @param newCreator java.lang.String
 */
public void setCreator(java.lang.String newCreator) {
   creator = newCreator;
}
/**
 * Insert the method's description here.
 * Creation date: (05-07-00 21:32:10)
 * @param newDescription java.lang.String
 */
public void setDescription(java.lang.String newDescription) {
   description = newDescription;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 17:40:00)
 * @param newDocuments java.util.Hashtable
 */
public void setDocuments(java.util.Hashtable newDocuments) {
   documents = newDocuments;
}
/**
 * Insert the method's description here.
 * Creation date: (05-07-00 21:25:46)
 * @param newGroupName java.lang.String
 */
public void setGroupName(java.lang.String newGroupName) {
   groupName = newGroupName;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 18:45:20)
 * @param newNumberOfDocuments int
 */
public void setNumberOfDocuments(int newNumberOfDocuments) {
   numberOfDocuments = newNumberOfDocuments;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 17:53:36)
 * @param newParticipants java.util.HashSet
 */
public void setParticipants(java.util.HashSet newParticipants) {
   participants = newParticipants;
}
/**
```

```
238   * Insert the method's description here.
      * Creation date: (07-07-00 20:11:38)
240   * @param newSessionURL com.sun.media.jsdt.URLString
      */
242  public void setSessionURL(com.sun.media.jsdt.URLString newSessionURL) {
        sessionURL = newSessionURL;
244  }
     /**
246   * Insert the method's description here.
      * Creation date: (08-07-00 20:06:01)
248   * @param newTotalDataVolume long
      */
250  public void setTotalDataVolume(long newTotalDataVolume) {
        totalDataVolume = newTotalDataVolume;
252  }
     }
```

Appendix D

PeerView help system

The following is a paper transscript of the online help system that ships with the PeerView binaries and is available as a separate World Wide Web document from [54]. Note that this material is intended for electronic publication and therefore may occasionally lend itself poorly to paper reproduction. It is included here for the sake of completeness and to serve as a user's manual for PeerView.

D.1 Introduction

Introduction

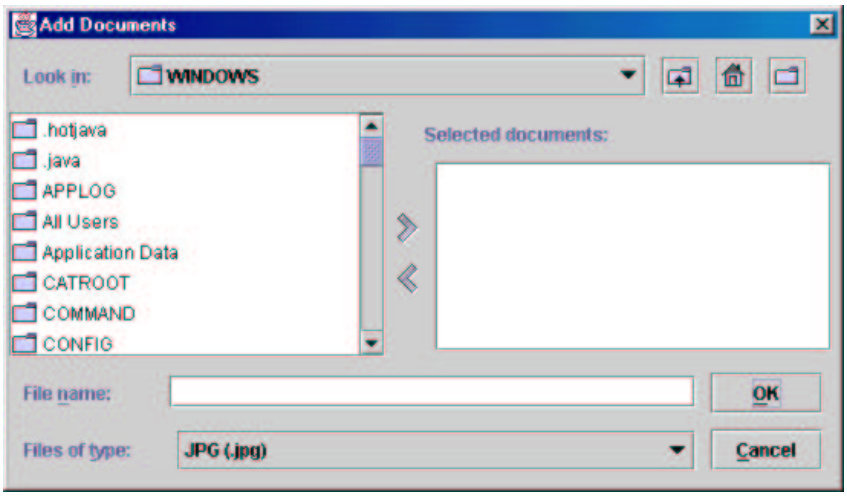
Welcome to the PeerView online help system.

PeerView is a system for artifact rendering and group review. It accomplishes its purpose through the use of a scalable, navigable panorama to which documents can be added and removed. Users of PeerView can join groups in order to share the documents in their panoramas. The documents can then be discussed by the group members using the discussion forum associated with each document. In addition, PeerView is customizable and can be configured to suit many different usage scenarios.

D.2 Add documents

Add documents

The "Add documents" box is used for adding documents to the panorama. You activate this box by selecting "Add documents" from the "File" menu or by pressing the button on the toolbar.



The features of the "Add documents" box are described below:

Look in

A listing of the drives that are visible from the location where your PeerView client was invoked. The active directory will be indicated by an expanded path from the root, i.e. the drive name, down to the directory itself.

The contents of the active directory, i.e. the files and subdirectories it contains, are displayed in the list box below the "Look in" label.

File name

The name of the file selected in the list box above the "File name" label. If multiple files are selected, this field will contain the name of the last file selected.

Files of type

A file filter indicating which type of file will be displayed in the list box above the "File name" label. Select this to exclude all other files than those relevant to your browsing.

Selected documents

A list of the documents selected from the leftmost list box. You transfer files to and from the right list box by first selecting one or more files in either box and then pressing one of the arrows between the list boxes to indicate the direction of transfer.

Buttons

Here's a brief legend to the buttons found in the "Add files" box:

Up one level. Makes the parent directory of the current directory active.

Home. Makes your home directory the active directory.

Create new folder. Creates a new folder in the active directory.

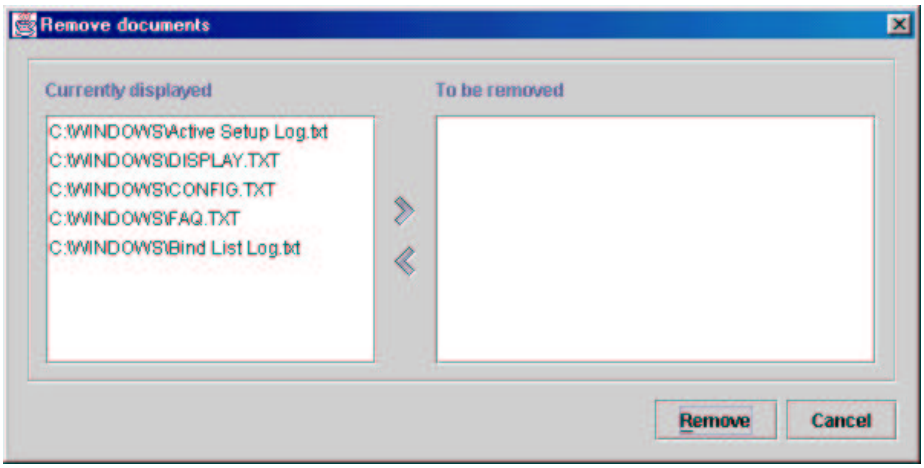
Click OK to add the files in the "Selected documents" list box to the panorama.

Click Cancel to close the "Add files" dialog without adding any documents to the panorama. Your selections, if any, will be discarded.

D.3 Remove documents

Remove documents

The "Remove documents" box is used for removing documents from the panorama. It can be activated by choosing "Remove documents" from the "File" menu or by pressing the button on the toolbar.



The features of the "Remove documents" box are described below:

Currently displayed

This list box shows which of your files are currently displayed in the panorama. Note that this does not include other group members' documents as these cannot be removed by you, only by the authors themselves. You move items from the "Currently displayed" box to the "To be removed" box by selecting one or more file names and pressing the button.

To be removed

This list box shows which of your files currently displayed in the panorama are to be removed. You move files from this box to the "Currently displayed" box selecting one or more and pressing the button.

Buttons

Remove

Close the remove box and remove all documents listed in the "To be removed" box from the panorama.

Cancel

Close the remove box without removing any documents from the panorama. Your selection, if any, will be discarded.

D.4 Panorama

Panorama

The panorama occupies the center portion of the PeerView client main window. It is where documents are placed for inspection, navigation and manipulation. The panorama is therefore usually the area where most interaction between you and the application takes place. For that same reason, the panorama can be controlled in a variety of ways as described below:

Resizing the panorama

You can adjust the size of the panorama relative to the other elements of the main window by dragging the divider bar.

Scaling the panorama

You can adjust the magnification of the panorama by scaling it up or down:

1) Position the mouse cursor on the panorama background, i.e. outside any document.

2) Press and hold the right mouse button.

3) Move the cursor left to increase magnification or right to decrease it.

4) Release the mouse button when you have reached the desired magnification.

Moving the panorama

You can move the panorama, or rather, its contents, using the mouse:

1) Position the mouse cursor on the panorama background, i.e. outside any document.

2) Press and hold the left mouse button.

3) Move the cursor to the panorama's new location.

4) Release the left mouse button.

Zooming to overview

You can get an overview of the contents of the panorama by zooming to overview. This will scale the panorama to a magnification that allows all of the contents to fit in the visible portion of the panorama:

1) Position the mouse cursor on the panorama background, i.e. outside any document

2) Double click the left mouse button

Activating a document

To activate a document, do the following:

1) Position the mouse cursor on the document in question.

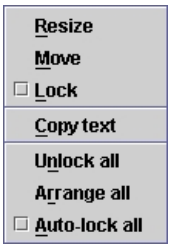
2) Double click the left mouse button

This will cause the PeerView application to center the document in the panorama while zooming to a magnification that allows the document to fill as much of the visible portion of the panorama as possible.

At the same time, the client will request the topic tree for that document if one is available, i.e. if there is a connection to an active PeerView server. When received by the client, the topic tree will be displayed in the discussion forum area at the bottom of the main window.

Document operations

You can carry out various operations that affect one or all of the documents in the panorama. This is achieved using the panorama pop−up menu:



To activate this menu and select an item from it:

1) Position the mouse cursor on any document.

2) Click the right mouse button.

3) Move the mouse cursor to the item you wish to select.

4) Click the left mouse button

Here is a brief explanation of each operation that can be selected from the menu:

Resize

After selecting this item, the lower right corner of the document's frame will "attach" itself to the mouse cursor and follow its movements around the panorama. To release the frame, press the left mouse button. The document will retain its new size until you resize it again or remove it from the panorama.

Move

After selecting this item, the document will "attach" itself to the mouse cursor by its upper left corner and follow its movements around the panorama. To release the document, press the left mouse button. The

document will remain at its new size until you move it again or, if its lock is disabled, until you arrange the documents.

Lock

If a document's lock is enabled, the document will remain at its position even if you select "Arrange all" from the pop−up menu. If the lock is disabled, the document will be repositioned at an appropriate location in the arrangement when you select "Arrange all".

Copy text

Copy any selected text to the clipboard. This item is disabled if the document does not contain text.

Unlock all

Disable all document locks.

Arrange all

Arrange any non−locked documents using the active layout manager. The panorama will be zoomed to overview after selecting this item.

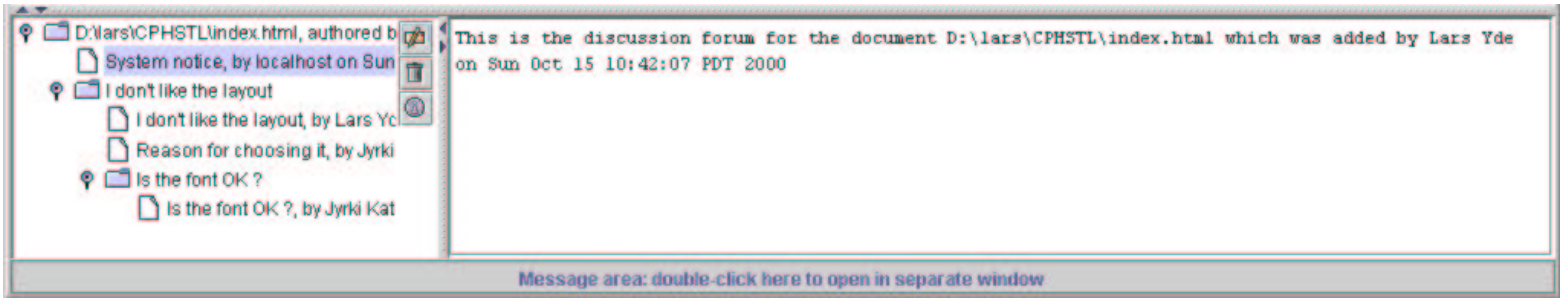
Auto−lock all

If this item is checked, any documents moved will have their lock enabled by default. Note that this does not affect documents moved prior to checking this item.

D.5 Discussion forum

Discussion forum

The discussion forum is the split pane display at the bottom of the PeerView main window. It is where discussion on documents take place and you can browse, add, delete and view messages using the controls in the forum.



The below describes the features of the discussion forum.

Topic tree

The structure to the left of the discussion forum which resembles the file system browsers available on many systems is called a "topic tree". It lists the "threads" of discussion, i.e. the topics that are being and have been discussed throughout the document's history as well as "messages", i.e. the individual contributions to the topics discussed.

The three small buttons in the top right corner of the discussion forum are used as follows:

Compose new message. To use this command, first select an item from the topic tree. If you select a topic, i.e. an item marked by the graphic, your message will be the first in a new subtopic to the selected topic. That subtopic will have the same title as your message so you should choose something that appropriately designates the entire thread of discussion you are about to initiate.

If the item you select is a message (an item marked by), your new message will be inserted immediately below the selected message.

Delete a message. You must select a message before clicking this button. If the message is authored by you and has not yet been responded to, you will be asked to confirm its deletion. If either of those conditions is not met, the message will not be deleted and you will therefore not be asked for confirmation. The reason for imposing this policy is to ensure that the topic tree reflects the actual flow of discussion and not an arbitrary sequence of additions and deletions.

View the properties of a selected message. First select an item, then click this button to view details about it such as author's name, contents, date of creation and so forth.

Message display

The message display is the rightmost portion of the discussion forum where messages are displayed. Each time you select a message from the topic tree, the client request the contents of that message from the server and displays them in the message display.

Adjusting the size of the discussion forum

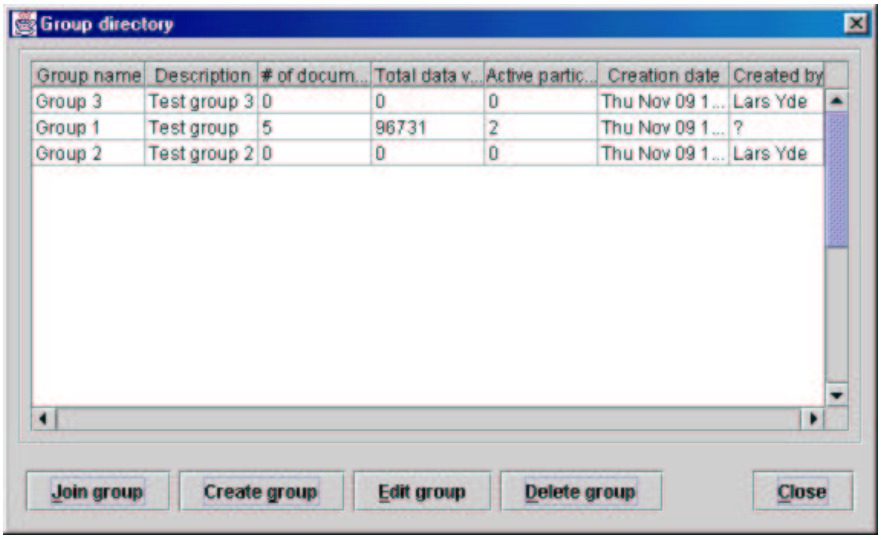
To adjust the size of the discussion forum itself, use the divider bar which separates it from the panorama.

To adjust the size of the topic tree and message display area, use the divider bar that separates the two.

D.6 Group directory

Group directory

The group directory box is used for joining, creating, deleting and editing groups. You activate the group directory by selecting "Groups" from the setup menu or by pressing the button on the toolbar.



The features of the group directory box are described below:

Group listing

The top portion of the group directory box is occupied by a listing of the groups available to you. This listing is requested from the server each time the group directory box is opened and is therefore always up−to−date. If for some reason the server is unavailable when the box is opened, the listing will be empty.

Each row in the listing gives the following information on a single group:

Group name

The name of the group.

Group description

A description of the group, entered when the group was created.

of documents

The total number of documents contributed to the group by its members.

Total data volume

The combined size of all documents in the group. This will give you an indication of how much network traffic you can expect when joining a given group.

Active participants

The number of group members currently taking part in the group.

Creation date

The date and time when the group was created.

Created by

The name of the person who created the group.

Resizing columns

You can adjust the width of the columns in the group listing. Place the mouse cursor over the indentation that separates two columns (). This should cause the cursor to change appearance and you can now drag the indentation to either side to resize the columns it separates.

Buttons

The following as a brief legend to the buttons available in the group directory box. Note that all of these button, except "Close" are disabled if there is no connection to the server since that makes it impossible to execute operations on the group listing.

Join the group selected in the group listing. A message box will appear if none is selected. Note that the group join operation may take a while to complete as a fair amount of network traffic may be generated when the other group members are requested to send their documents to you and vice versa.

Create a new group. You will be asked to give the name and description of the group after pressing this button. The new group will appear in the group listing.

If a group has been selected from the group listing, you will be asked to edit its name and description and confirm/discard your changes. If no group is selected, a message box to that effect will appear.

If a group has been selected from the group listing, you will be asked to confirm or cancel its deletion. If no group has been selected, a message box to that effect will appear.

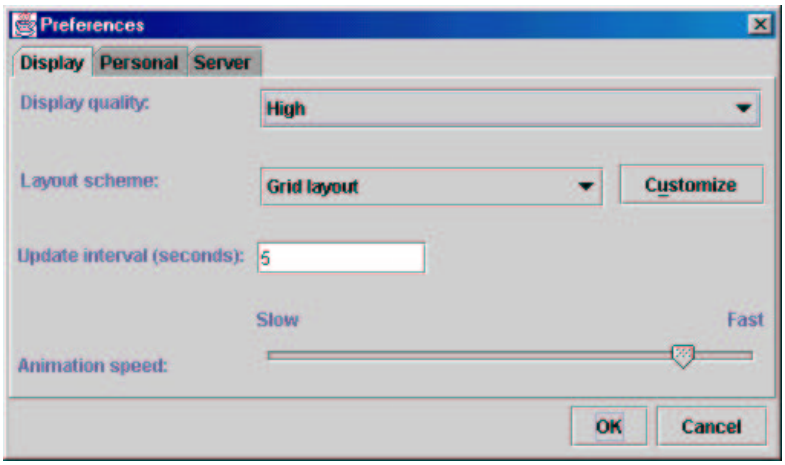
Close the group directory box.

D.7 Preferences

Preferences

The preferences box is used for controlling the behaviour of the PeerView application. You activate the preferences box by selecting "Preferences" from the "Setup" menu or by pressing the button on the toolbar.

Display setup



The display setup pane is used for configuring the behaviour of the PeerView panorama. The following can be controlled using the items on this pane:

Display quality

The quality in which documents are rendered. The higher the quality, the better the resolution and appearance of the documents in the panorama but also, the lower the speed at which the panorama can be navigated and scaled.

Layout scheme

The algorithm, i.e. method, used to layout the documents in the panorama. Press the "Customize" button to adjust the settings of the layout scheme selected from the drop−down box.

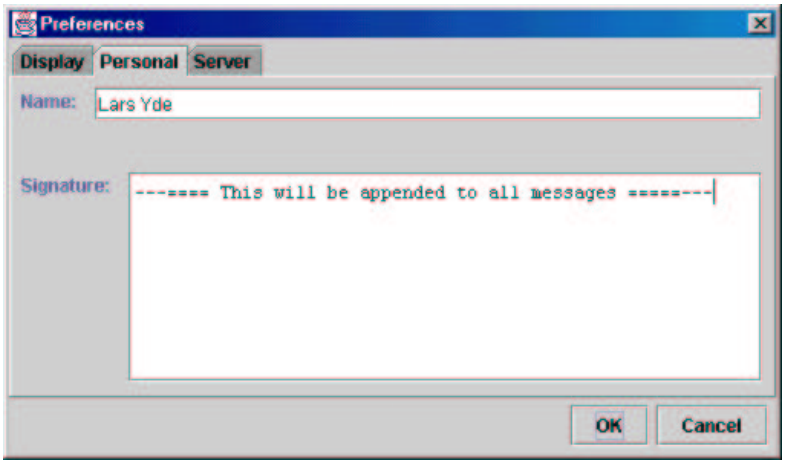
Update interval

The interval at which PeerView will check the files you have added to the panorama for changes. Any changes will be broadcast across the network to the other group members. The lower this setting, the more up−to−date documents you can expect other group members to have. However, low settings can also cause a noticeable load on your machine and network.

Animation speed

The speed at which zoom operations on the panorama are performed.

Personal setup



The "Personal" pane is used for setting properties that identify you to other users. The following items are available:

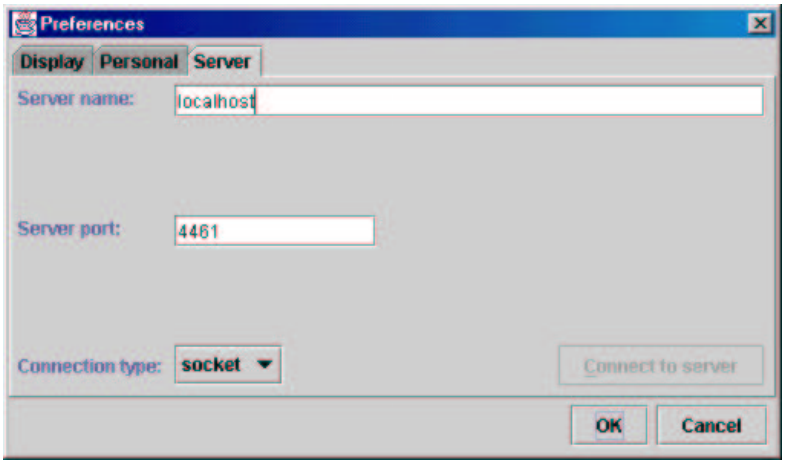
Name

Your name. This needn't be your real name, any pseudonym will do, but do not change it too often since any messages composed by you will be identifiable to others by your name only.

Signature

A signature text that will be automatically appended to all messages you compose.

Server setup



The "Server" pane is used for setting the parameters used when connecting to the server at startup. The following items are available:

Server name

The name of the server you wish to connect to at startup. Enter "localhost" as in the image above to connect to a server running locally, i.e. on the machine you are running the PeerView client from. If you are connecting to a remove server, enter its DNS address, not its IP, as in:

www.myserver.myinstitution.edu

Server port

The server port that the PeerView client should connect to on startup.

IMPORTANT NOTE: You must ensure that this port is open on the server machine, i.e. exempt from any firewall protection or PeerView's connection attempt will fail.

Also, **port number 4561 must always be open on the server machine**, regardless of the port number you enter in this field. PeerView thus requires (at least) two ports to be open and accessible at the server end.

Connection type

The type of connection protocol to use. For connections over the internet, you should use a HTTP protocol. For LAN and localhost uses, you should probably select socket, although HTTP may also apply on some networks.

Connect to server

This feature is not available in the beta release (#0.900) of PeerView.

Buttons

Press "OK" to close the preferences box and save your changes.

Press "Cancel" to close preferences box without saving your changes.

D.8 Using a divider bar

Using a divider bar

A horizontal divider bar looks like this:

 (appearance may be slightly different on your platform)

and is used for adjusting the size of the areas adjacent to it. To use a divider bar, place the mouse cursor over it, press the left mouse button and drag the bar up or down while holding the button. Alternatively, you can press either of the two arrow symbols at the left end of the divider bar. Pressing the upwards pointing arrow will minimize the area above the divider while pressing the downwards pointing arrow will maximize it.

The above instructions also apply to vertical divider bars although the bar should of course be dragged horizontally rather than vertically.

Bibliography

[1] P. Young (Editor), *3D Software Visualization — Visualisations and Representations*, Worldwide Web Document (1997). Available at `http://vrg.dur.ac.uk/misc/PeterYoung/pages/work/documents/BT-report/`.

[2] J. Arvo (Editor), *Graphics Gems II*, Academic Press, Inc. (1991).

[3] K. Baker, S. Greenberg, and C. Gutwin, *Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration*, Technical report, Department of Computer Science, University of Calgary (2000).

[4] L. Bannon and S.B. dker, Constructing Common Information Spaces, *Proceedings of ECSCW97, Dordrecht*, Kluwer (1997), 81–96.

[5] P.L.P.M. Barbara Hayes-Roth,Karl Pfeger and M. Balabanovic, A Domain-Specific Software Architecture for Adaptive Intelligent Systems, *IEEE Transactions on Software Engineering* **21**,4 (1995).

[6] K. Beck, Extreme Programming Explained (2000).

[7] B. Bederson and J. Meyer, Implementing a zooming User Interface: experience building Pad++, *Software Practice and Experience* **28**,10 (1998), 1101–1135.

[8] B.B. Bederson and B. McAlister, *Jazz: An Extensible 2D+Zooming Graphics Toolkit in Java*, Technical Report CS-TR-4015, University of Maryland, College Park (1999).

[9] A.F. Blackwell, *Metaphor in Diagrams*, Ph.D. Thesis, Darwin College, University of Cambridge (1998).

[10] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, 1 Edition, Addison-Wesley, Reading, Massachusetts, USA (1999).

[11] F.P. Brooks, No Silver Bullet - Essence and Accidents of Software Engineering, *IEEE Computer* (1987), 10–19.

[12] Centrinity Inc., *Centrinity*, Website accessible at `http://www.softarc.com`.

[13] www.cryptix.org, Website accessible at `http://www.cryptix.org`.

[14] OpenAvenue, Inc., *Concurrent Versions System: The Open Standard for Version Control*, Website accessible at `http://www.cvshome.org`.

[15] Cybozu Inc., *Cybozu*, Website accessible at `http://www.cybozu.com`.

[16] L.P. Deutsch, *RFC 1951: DEFLATE Compressed Data Format Specification version 1.3* (1996). Status: INFORMATIONAL.

[17] P. Dourish and S. Bly, Portholes: Supporting Awareness in a Distributed Work Group, *ACM Annual Conference on Human Factors in Computing Systems* (1992).

[18] Lehrgebiet Praktische Informatik II, FernUniversität Hagen, *DreamTeam Homepage*, Website accessible at `http://carmen.fernuni-hagen.de/dreamteam/dreamteam_eng.html`.

[19] S. Easterbrook, *Coordination Breakdowns: Why Groupware is so Difficult to Design*, Technical Report CSRP 343, School of Cognitive and Computing Sciences, University of Sussex.

[20] G. W. Furnas and X. Zhang, MuSE: A Multiscale Editor, *Proceedings of the ACM Symposium on User Interface Software and Technology, Zoomable User Interfaces* (1998), 107–116.

[21] R. L. Glass, *Software Runaways: Monumental Software Disasters*, Prentice Hall, Upper Saddle River (1998).

[22] S. Greenberg, C. Gutwin, and A. Cockburn, Awareness Through Fisheye Views in Relaxed-WYSIWIS Groupware, *Graphics Interface '96*, W. A. Davis and R. Bartels (Editors), Canadian Human-Computer Communications Society (1996), 28–38. ISBN 0-9695338-5-3.

[23] R. E. Grintner, Workflow Systems: Occasions for Success and Failure, *Computer Supported Cooperative Work* **9** (2000), 189–214.

[24] Department of Computer Science, University of Calgary, *GroupLab: Laboratory for Computer Supported Cooperative Work & Human Computer Interaction*, Website accessible at `http://www.cpsc.ucalgary.ca/projects/grouplab`.

[25] Diamond Bullet Design, *Types of Groupware and Groupware Issues*, Worldwide Web Document (2001). Available at `http://www.usabilityfirst.com/cscw.html#TypesofGroupware`.

[26] C. Gutwin and S. Greenberg, Effects of Awareness Support on Groupware Usability, *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems, Usability of Groupware* **1** (1998), 511–518.

[27] C. Gutwin and S. Greenberg, *A Framework of Awareness for Small Groups in Shared-Workspace Groupware*, Technical Report 99-1, Department of Computer Science, University of Saskatchewan (1999).

[28] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, *NCSA Habanero®*, Website accessible at `http://havefun.ncsa.uiuc.edu/habanero/`.

[29] D. T. Hoang, *Fast and Efficient Algorithms for Text and Video Compression*, Technical Report CS-97-06, Brown University - Department of Computer Science (1997).

[30] W. S. Humphrey, A Personal Commitment to Software Quality (1995), 5–7.

[31] J. J. Hunt, K. P. Vo, and W. F. Tichy, An Empirical Study of Delta Algorithms, *Software configuration management: ICSE 96 SCM-6 Workshop*, I. Sommerville (Editor), Springer (1996), 49–66.

[32] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley (1999).

[33] I. Jacobson and others, *Object-Oriented Software Engineering — a Use Case Driven Approach*, Addison-Wesley (1992).

[34] Human-Computer Interaction Lab, University of Maryland, *Welcome to Jazz!*, Website accessible at `http://www.cs.umd.edu/hcil/jazz/`.

[35] C. A. Jones, The Java Internationalization API: Global software for the global village, *Dr. Dobb's Journal of Software Tools* **23**,1 (1998), 54, 56–69, 103–104.

[36] Sun Microsystems, Inc., *Java™ Shared Data Toolkit*, Website accessible at `http://java.sun.com/products/java-media/jsdt/index.html`.

[37] T. Liao, *Light-weight Reliable Multicast Protocol*, Technical report, INRIA (1998).

[38] LOTUS DEVELOPMENT CORPORATION, *Lotus Notes*, Website accessible at `http://www.lotus.com/home.nsf/welcome/notes`.

[39] INRIA, *LRMP Home Page*, Website accessible at `http://webcanal.inria.fr/index.html`.

[40] S. MCCONNELL, *Code Complete*, Microsoft Press (1990).

[41] B. MEYER, Component and Object Technology: On to Components, *Computer* **32**,1 (1999), 139–140.

[42] H. D. MILLS, M. DYER, AND R. LINGER, Cleanroom Software Engineering, *IEEE Software* **4**,5 (1987), 19–25.

[43] J. NIELSEN, *Usability Engineering*, Academic Press, Inc. (1993).

[44] WWW.OPENGL.ORG, *Open GL*, Website accessible at `http://www.opengl.org/`.

[45] D. PINELLE, *A Survey of Groupware Evaluations in CSCW Proceedings*, Technical report, Department of Computer Science, University of Saskatchewan (2000).

[46] M. ROSEMAN AND S. GREENBERG, TeamRooms: Network Places for Collaboration (1996), 325–333.

[47] C. G. SAUL GREENBERG AND A. COCKBURN, Using Distortion-Oriented Displays to Support Workspace Awareness, R. C. A.SASSE AND R.WINDER (Editors), Springer-Verlag (1996), 299–314.

[48] K. SCHMIDT AND L. BANNON, Taking CSCW seriously: Supporting articulation work, *Computer Supported Cooperative Work* **1** (1992), 7–40.

[49] B. SCHNEIDERMAN, *Designing the User Interface*, Addison-Wesley (1998).

[50] SUN MICROSYSTEMS, INC., *Java™ 2 SDK, Standard Edition*, Website accessible at `http://java.sun.com/products/jdk/1.2/`.

[51] I. SOMMERVILLE, *Software Engineering*, 5th Edition, Addison-Wesley Publishing Company, Inc. (1996).

[52] J. STASKO, J. DOMINGUE, M. H. BROWN, AND B. A. PRICE (Editors), *Software Visualization: Programming as a Multimedia Experience*, The MIT Press (1998).

[53] WEBWISDOM.COM, INC., *TANGO Interactive™*, Website accessible at `http://www.webwisdom.com/tangointeractive`.

[54] LARS YDE, *The PeerView Website*, Website accessible at `http://www.diku.dk/research-groups/performance-engineering/PeerView/`.

[55] E. TUFTE, *Envisioning Information*, Graphics Press (1990).

[56] DIAMOND BULLET DESIGN, *Usability First™*, Website accessible at `http://www.usabilityfirst.com/`.

[57] WEBMAGIC, INC., *USENET.org™*, Website accessible at `http://www.usenet.org`.

[58] IBM, *VisualAge*, Worldwide Web Document. Available at `http://www-4.ibm.com/software/ad/vajava/`.

[59] A. WOODRUFF, J. LANDAY, AND M. STONEBRAKER, Goal-Directed Zoom, *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems (Summary), Late Breaking Results: Look and Learn: Visualization and Education Too* **2** (1998), 305–306.

[60] DISTRIBUTED SYSTEMS TECHNOLOGY CENTRE PTY. LTD., *The wOrlds Project*, Website accessible at `http://archive.dstc.edu.au/TU/wOrlds/`.

[61] L. YDE, PeerView - a Prototype for Rendering and Review, *Distributed Systems Online* (2001).

[62] L. Yde and J. Katajainen, *Supporting Intellectual Work Through Artifact Rendering and Group Review*, Technical Report 00/11, Department of Computer Science, University of Copenhagen (DIKU) (2001).