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`

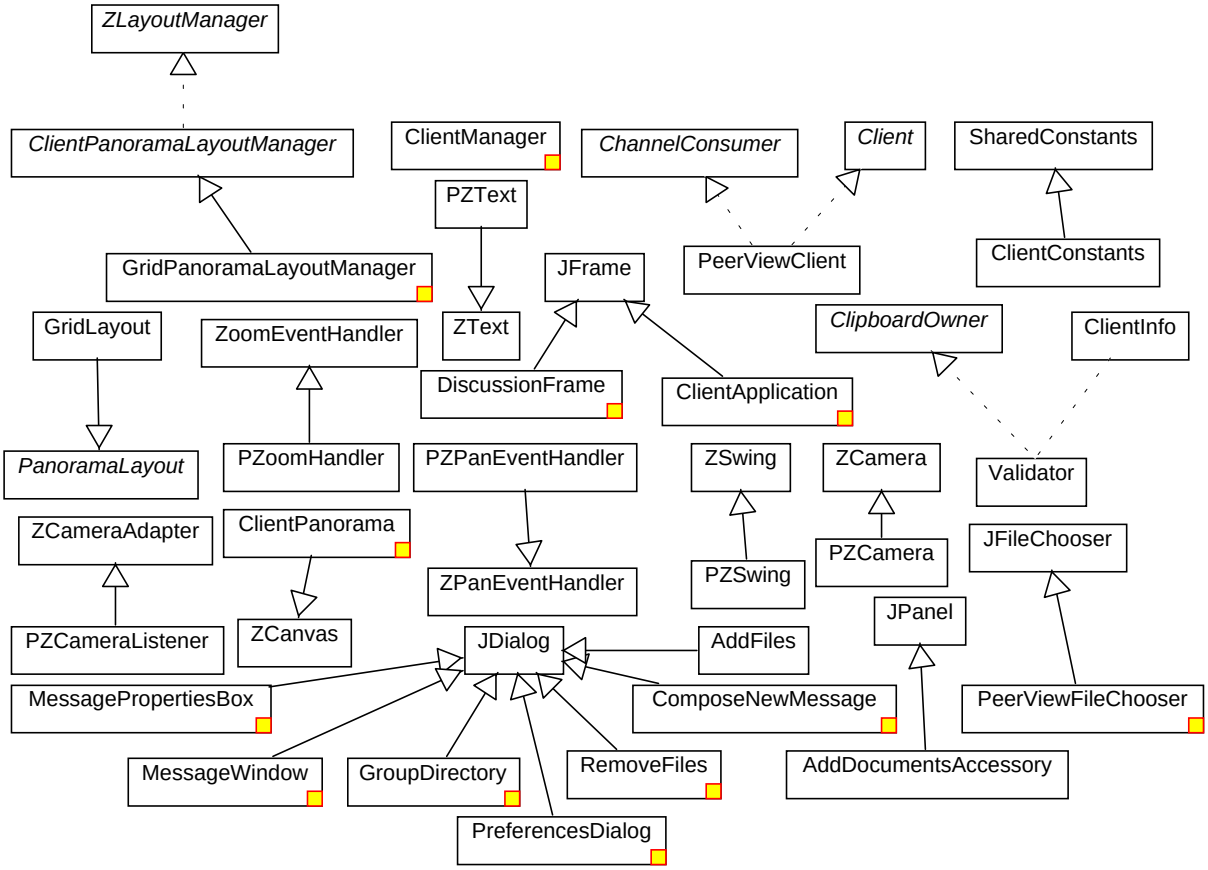


Figure A.1: Class diagram for the PeerView client application classes.

A.4 Class diagram for package `serverapp`

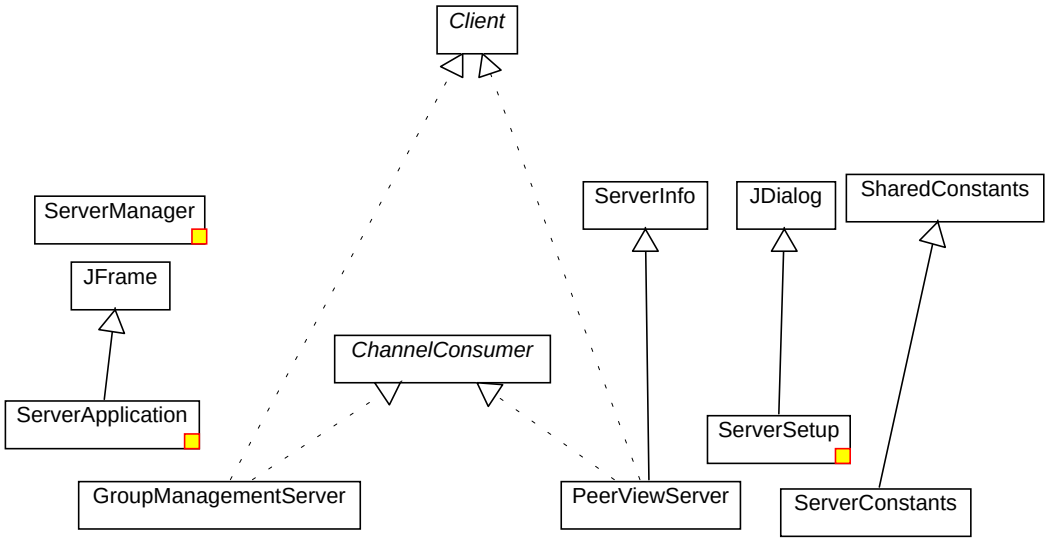


Figure A.2: Class diagram for the PeerView server application classes.

A.5 Class diagram for package `peerviewmisc`

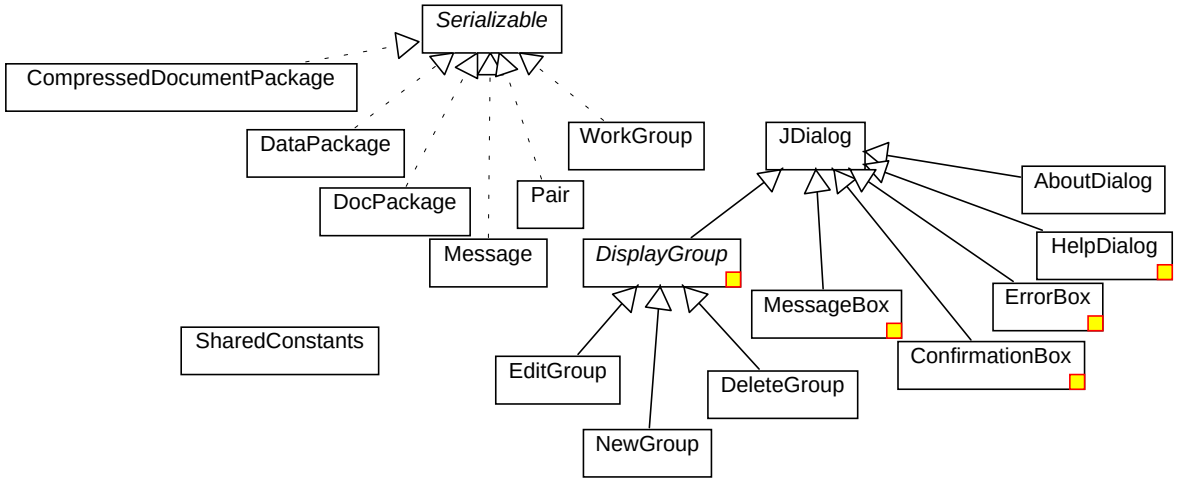


Figure A.3: Class diagram for shared and miscellaneous classes.

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 hovedvinducts 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 sĭdt 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 ndu 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 sluppet 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 DELS, 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 synes 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

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;
```

```
26  /**
     * AddDocumentsAccessory constructor comment.
28   */
    public AddDocumentsAccessory() {
30     super();
       initialize();
32  }
    /**
34   * AddDocumentsAccessory constructor comment.
     * @param layout java.awt.LayoutManager
36   */
    public AddDocumentsAccessory(java.awt.LayoutManager layout) {
38     super(layout);
    }
40  /**
     * AddDocumentsAccessory constructor comment.
42   * @param layout java.awt.LayoutManager
     * @param isDoubleBuffered boolean
44   */
    public AddDocumentsAccessory(java.awt.LayoutManager layout, boolean isDoubleBuffered) {
46     super(layout, isDoubleBuffered);
    }
48  /**
     * AddDocumentsAccessory constructor comment.
50   * @param isDoubleBuffered boolean
     */
52  public AddDocumentsAccessory(boolean isDoubleBuffered) {
       super(isDoubleBuffered);
54  }
    /**
56   * Return the AddButton property value.
     * @return javax.swing.JButton
58   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
60  public javax.swing.JButton getAddButton() {
       if (ivjAddButton == null) {
62        try {
             ivjAddButton = new javax.swing.JButton();
64           ivjAddButton.setName("AddButton");
             ivjAddButton.setToolTipText("Add␣to␣list");
66           ivjAddButton.setMnemonic('a');
             ivjAddButton.setText("");
68           ivjAddButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                  navigation/Forward24.gif")));
             ivjAddButton.setBorderPainted(false);
70           ivjAddButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
72           // user code end
          } catch (java.lang.Throwable ivjExc) {
74           // user code begin {2}
             // user code end
76           handleException(ivjExc);
          }
78     }
       return ivjAddButton;
80  }
    /**
82   * Return the ButtonPanel property value.
     * @return javax.swing.JPanel
84   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
86  private javax.swing.JPanel getButtonPanel() {
       if (ivjButtonPanel == null) {
88        try {
             ivjButtonPanel = new javax.swing.JPanel();
90           ivjButtonPanel.setName("ButtonPanel");
```

```
            ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
92          ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
            ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
94
            java.awt.GridBagConstraints constraintsAddButton = new java.awt.GridBagConstraints();
96          constraintsAddButton.gridx = 0; constraintsAddButton.gridy = 0;
            getButtonPanel().add(getAddButton(), constraintsAddButton);
98
            java.awt.GridBagConstraints constraintsRemoveButton = new java.awt.GridBagConstraints();
100         constraintsRemoveButton.gridx = 0; constraintsRemoveButton.gridy = 1;
            getButtonPanel().add(getRemoveButton(), constraintsRemoveButton);
102         // user code begin {1}
            // user code end
104      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
106         // user code end
            handleException(ivjExc);
108      }
      }
110   return ivjButtonPanel;
   }
112 /**
    * Return the DocumentList property value.
114 * @return javax.swing.JList
    */
116 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   public javax.swing.JList getDocumentList() {
118   if (ivjDocumentList == null) {
         try {
120         ivjDocumentList = new javax.swing.JList();
            ivjDocumentList.setName("DocumentList");
122         ivjDocumentList.setToolTipText("Documents␣displayed␣in␣panorama");
            ivjDocumentList.setBounds(0, 0, 112, 253);
124         // user code begin {1}
            selectedDocuments = new DefaultListModel();
126         ivjDocumentList.setModel(selectedDocuments);
            // user code end
128      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
130         // user code end
            handleException(ivjExc);
132      }
      }
134   return ivjDocumentList;
   }
136 /**
    * Return the JLabel property value.
138 * @return javax.swing.JLabel
    */
140 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getJLabel() {
142   if (ivjJLabel == null) {
         try {
144         ivjJLabel = new javax.swing.JLabel();
            ivjJLabel.setName("JLabel");
146         ivjJLabel.setText("␣␣Selected␣documents:");
            // user code begin {1}
148         // user code end
         } catch (java.lang.Throwable ivjExc) {
150         // user code begin {2}
            // user code end
152         handleException(ivjExc);
         }
154   }
      return ivjJLabel;
156 }
```

```
    /**
158  * Return the JScrollPane property value.
     * @return javax.swing.JScrollPane
160  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
162 private javax.swing.JScrollPane getJScrollPane() {
       if (ivjJScrollPane == null) {
164       try {
             ivjJScrollPane = new javax.swing.JScrollPane();
166          ivjJScrollPane.setName("JScrollPane");
             getJScrollPane().setViewportView(getDocumentList());
168          // user code begin {1}
             // user code end
170       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
172          // user code end
             handleException(ivjExc);
174       }
       }
176    return ivjJScrollPane;
    }
178 /**
     * Return the RemoveButton property value.
180  * @return javax.swing.JButton
     */
182 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    public javax.swing.JButton getRemoveButton() {
184    if (ivjRemoveButton == null) {
          try {
186          ivjRemoveButton = new javax.swing.JButton();
             ivjRemoveButton.setName("RemoveButton");
188          ivjRemoveButton.setToolTipText("Remove␣from␣list");
             ivjRemoveButton.setMnemonic('r');
190          ivjRemoveButton.setText("");
             ivjRemoveButton.setMaximumSize(new java.awt.Dimension(29, 35));
192          ivjRemoveButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics
                 /navigation/Back24.gif")));
             ivjRemoveButton.setBorderPainted(false);
194          ivjRemoveButton.setPreferredSize(new java.awt.Dimension(29, 35));
             ivjRemoveButton.setMinimumSize(new java.awt.Dimension(29, 35));
196          ivjRemoveButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
198          // user code end
          } catch (java.lang.Throwable ivjExc) {
200          // user code begin {2}
             // user code end
202          handleException(ivjExc);
          }
204    }
       return ivjRemoveButton;
206 }
    /**
208  * Insert the method's description here.
     * Creation date: (15-06-00 16:07:02)
210  * @return javax.swing.DefaultListModel
     */
212 public javax.swing.DefaultListModel getSelectedDocuments() {
       return selectedDocuments;
214 }
    /**
216  * Called whenever the part throws an exception.
     * @param exception java.lang.Throwable
218  */
    private void handleException(java.lang.Throwable exception) {
220
       /* 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     }
     }
```

```
288 /**
     * Set the selectedDocuments property
290  * Creation date: (15-06-00 16:07:02)
     * @param newSelectedDocuments javax.swing.DefaultListModel
292  */
    public void setSelectedDocuments(javax.swing.DefaultListModel newSelectedDocuments) {
294    selectedDocuments = newSelectedDocuments;
    }
296 }
```

Listing C.2: AddFiles.java

```
    package clientapp;
 2
    /**
 4   * The document used for adding files/documents to the panorama.
     * Creation date: (23-05-00 08:55:51)
 6   * @author:
     */
 8  import javax.swing.*;
    import java.awt.*;
10  import java.io.*;
    import java.awt.event.*;
12  import javax.swing.filechooser.FileFilter;
    import java.io.File.*;
14
    public class AddFiles extends JDialog {
16
       private JPanel ivjJDialogContentPane = null;
18     private PeerViewFileChooser ivjFileChooser = null;
       private ClientApplication clientApplication;
20  /**
     * Constructor
22   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
24  public AddFiles() {
       super();
26     initialize();
    }
28  /**
     * AddFiles constructor comment.
30   * @param owner java.awt.Dialog
     */
32  public AddFiles(java.awt.Dialog owner) {
       super(owner);
34     initialize();
    }
36  /**
     * AddFiles constructor comment.
38   * @param owner java.awt.Dialog
     * @param title java.lang.String
40   */
    public AddFiles(java.awt.Dialog owner, String title) {
42     super(owner, title);
    }
44  /**
     * AddFiles constructor comment.
46   * @param owner java.awt.Dialog
     * @param title java.lang.String
48   * @param modal boolean
     */
50  public AddFiles(java.awt.Dialog owner, String title, boolean modal) {
       super(owner, title, modal);
52  }
    /**
54   * 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) {
188      handleException(ivjExc);
      }
190   // user code begin {2}
      ivjFileChooser.setAddFilesDialog(this);
192   // default positioning puts the dialog at center of screen. The method used is a slight hack, but
          works fine.
      setLocationRelativeTo( getOwner() );
194   // user code end
   }
196 /**
    * main entrypoint - starts the part when it is run as an application
198 * @param args java.lang.String[]
    */
200 public static void main(java.lang.String[] args) {
      try {
202      AddFiles aAddFiles;
         aAddFiles = new AddFiles((JFrame)null);
204      aAddFiles.setModal(true);
         aAddFiles.addWindowListener(new java.awt.event.WindowAdapter() {
206         public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
208         };
         });
210      aAddFiles.setVisible(true);
      } catch (Throwable exception) {
212      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
214   }
   }
216 /**
    * Insert the method's description here.
218 * Creation date: (16-06-00 08:34:47)
    * @param newClientApplication clientapp.ClientApplication
220 */
   public void setClientApplication(ClientApplication newClientApplication) {
222   clientApplication = newClientApplication;
   }
224 /**
    * Overrides the JDialog show method to ensure that init is called before the box is displayed.
226 * Creation date: (03-08-00 01:22:58)
    */
228 public void show()
   {
230   getFileChooser().init();
      super.show();
232 }
   /**
234 * Add the files selected by the user to the panorama and dispose of the dialog box.
    * Creation date: (12-06-00 07:47:56)
236 */
   public void submitAndClose()
238 {
      Object selectedObjects[] = getFileChooser().getFileChooserAccessory().getSelectedDocuments().toArray
          ();
240   File selectedFiles[] = new File[selectedObjects.length];
      for (int i = 0; i < selectedFiles.length; i++)
242   {  selectedFiles[i] = (File)selectedObjects[i]; }
      clientApplication.getClientManager().addDocuments(selectedFiles);
244   clientApplication.repaint();
      dispose();
246 }
   }
```

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") ));
186         renderLabel.setForeground( ClientConstants.getVISIBLE_DOCUMENTS_SELECTION_COLOUR() );
         }
188      renderLabel.setText( ClientConstants.getVISIBLE_DOCUMENTS_TAB_SIZE() +
                docName );
190

192      }
         // caption item last since it is the least likely (being unique)
194      else if ( value instanceof CaptionItem )
         {
196         renderLabel.setText( ( (CaptionItem) value ).getCaption() );
         // renderLabel.setIcon( new javax.swing.ImageIcon( getClass().getResource( "/
              toolbarButtonGraphics/navigation/Home16.gif" )));
198      renderLabel.setHorizontalAlignment( SwingConstants.CENTER );
         // renderLabel.setBorder( BorderFactory.createMatteBorder( 6, 6, 6, 6, renderLabel.getBackground
              () ));
200      }
         return renderLabel;
202   }
   };
204
   /** Event handler for the message area at the bottom of the client application. */
206 public class MessageAreaMouseListener extends MouseAdapter
   {
208   public void mouseClicked( MouseEvent me )
      {
210      {
         if ( me.getClickCount() == 2 )
212      {
            showMessageWindow();
214      }
         }
216   }
   }
218
   /** Event handler for the client application main window */
220 public class ClientWindowListener extends WindowAdapter
   {
222   public void windowActivated( WindowEvent we )
      {
224      bringChildWindowsOnTop();
      }
226   public void windowClosing( WindowEvent we )
      {
228       //System.out.println(" windowClosing ");
         try
230      {
            terminateAndClose();
232         dispose();
         }
234      catch ( Throwable t )
         {
236      }
      }
238 }
   /** Event handler for the topic tree area of the client application main windows */
240 public class TopicTreeSelectionListener implements TreeSelectionListener
   {
242   public void valueChanged( TreeSelectionEvent tse )
      {
244      handleTopicTreeClick( tse.getPath() );
      }
246 }
```

```
248     /** 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.
250     */
        public class Task implements ActionListener
252     {
           private int currentValue;
254
           public Task(int startValue)
256        {
              currentValue = startValue;
258        }

260        public void actionPerformed(ActionEvent actionEvent)
           {
262           updateMessageArea(currentValue);
              // System.out.println("ACTION!");
264        }

266        public void updateValue(int newValue)
           {
268           currentValue = newValue;
           }
270
           public void advance()
272        {
              currentValue++;
274        }
        }
276
        /** This class is used to represent clients that participate in JSDT sessions. It therefore
            implements the JSDT Client interface.
278     */
        public class DefaultClient implements Client
280     {
           String name;
282
           public DefaultClient (String nameArg )
284        {
              name = nameArg;
286        }
           public Object authenticate ( AuthenticationInfo ai )
288        {
              return null;
290        }
           public String getName ()
292        {
              return name;
294        }
        }
296
        public class TopicTreeMouseAdapter extends MouseAdapter
298     {
           public void mouseClicked( MouseEvent me )
300        {
           // handleTopicTreeClick( me );
302        }
        }
304
        /** The renderer class for the discussion tree in the client application main window. Leaf nodes in
            the tree are
306     * rendered with the string representation of the user object embedded in the node
        */
308     public class DiscussionTreeCellRenderer extends DefaultTreeCellRenderer
        {
310
```

```
      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.
```

```
438   */
      public ClientApplication() {
440      super();
         initialize();
442   }
      /**
444    * ClientApplication constructor comment.
       * @param title java.lang.String
446    */
      public ClientApplication(String title) {
448      super(title);
      }
450   /**
       * Add a new item to the list of visible documents found in the toolbar panel of the client application.
452    * Creation date: (30-08-00 15:41:37)
       * @param senderName java.lang.String
454    * @param authorName java.lang.String
       * @param documentName java.lang.String
456    */
      public void addDocumentToVisibleList(String senderName, String authorName, String documentNameAndPath,
           String documentName)
458   {
         String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
460      if ( getVisibleDocumentsList().containsKey( senderName ))
         {
462         ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
            docList.add( new DocumentItem( fqPath, documentName ));
464      }
         else
466      {
            ArrayList docList = new ArrayList();
468         docList.add( new AuthorItem( authorName ));
            docList.add( new DocumentItem( fqPath, documentName ));
470         getVisibleDocumentsList().put( senderName, docList );
         }
472   }
      /**
474    * Advance the progress bar at the bottom of the client application by one step.
       * Creation date: (19-06-00 11:04:10)
476    */
      public void advanceTask()
478   {
         if ( ivjMessageArea.getValue() >= ivjMessageArea.getMaximum() )
480      {
            initMessageAreaTimer();
482         updateMessageArea( ivjMessageArea.getValue() );
         }
484      else
         {
486         updateMessageArea( ivjMessageArea.getValue()+1 );
         }
488   }
      /**
490    * 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.
492    * Creation date: (16-08-00 15:32:07)
       */
494   public void bringChildWindowsOnTop()
      {
496      for ( int i = 0; i < getOwnedWindows().length; i++ )
         {
498         getOwnedWindows()[i].toFront();
         }
500   }
      /**
```

```
502   * Center and size the window at predefined dimensions and position relative to the screen resolution
      * Creation date: (05-08-00 19:20:24)
504   */
      public void center()
506   {

508      Dimension size;
         Dimension preferredSize = new Dimension(
510                  Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                         getFRAME_HORIZONTAL_SIZE() )),
                     Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                         getFRAME_VERTICAL_SIZE() )));
512      Dimension defaultSize = new Dimension(
                     Integer.parseInt( ClientConstants.getDEFAULT_FRAME_HORIZONTAL_SIZE() ),
514                  Integer.parseInt( ClientConstants.getDEFAULT_FRAME_VERTICAL_SIZE() ));
         Point position;
516      Point preferredPosition = new Point(
                     Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                         getFRAME_X_COORDINATE() )),
518                  Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
                         getFRAME_Y_COORDINATE() )));
         Point defaultPosition = new Point(
520                  Integer.parseInt( ClientConstants.getDEFAULT_FRAME_X_COORDINATE() ),
                     Integer.parseInt( ClientConstants.getDEFAULT_FRAME_Y_COORDINATE() ));
522      // if no value has overridden the default setting then query the OS for the dimensions of the display
         if ( preferredSize.equals( defaultSize ))
524      {
            size = new Dimension(  (int) (getToolkit().getScreenSize().getWidth() * ClientConstants.
                getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
526                        (int) (getToolkit().getScreenSize().getHeight() * ClientConstants.
                               getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) );
         }
528      else
         {
530         size = preferredSize;
         }
532      if ( preferredPosition.equals( defaultPosition ))
         {
534         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) ));
536      }
         else
538      {
            position = preferredPosition;
540      }
         // System.out.println( size.toString() );
542      setSize( size );
         setLocation( position );
544      setState( java.awt.Frame.NORMAL );
         repaint();
546   }
      /**
548   * 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.
550   * Creation date: (10-07-00 22:00:41)
      */
552   public void composeNewMessage()
      {
554      if ( getClientManager().verifyGroupConnectionManagement() == false )
         {
556         return;
         }
558      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 }
```

```
   /**
874  * Insert the method's description here.
     * Creation date: (28-07-00 00:27:12)
876  */
   protected void finalize() throws Throwable
878 {
      // System.out.println( " finalized ") ;
880   // ensure proper termination
   /* if ( isTerminated() == false )
882   {
        terminateAndClose();
884   }
   */
886 }
   /**
888  * Return the About property value.
     * @return javax.swing.JMenuItem
890  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
892 private javax.swing.JMenuItem getAbout() {
      if (ivjAbout == null) {
894      try {
           ivjAbout = new javax.swing.JMenuItem();
896        ivjAbout.setName("About");
           ivjAbout.setText("About");
898        // user code begin {1}
           // user code end
900      } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
902        // user code end
           handleException(ivjExc);
904      }
      }
906   return ivjAbout;
   }
908 /**
     * Return the AboutDialog1 property value.
910  * @return clientapp.AboutDialog
     */
912 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private AboutDialog getAboutDialog1() {
914   if (ivjAboutDialog1 == null) {
         try {
916        ivjAboutDialog1 = new peerviewmisc.AboutDialog();
           ivjAboutDialog1.setName("AboutDialog1");
918        ivjAboutDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           // user code begin {1}
920        // user code end
         } catch (java.lang.Throwable ivjExc) {
922        // user code begin {2}
           // user code end
924        handleException(ivjExc);
         }
926   }
      return ivjAboutDialog1;
928 }
   /**
930  * Return the AddDocuments property value.
     * @return javax.swing.JMenuItem
932  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
934 public javax.swing.JMenuItem getAddDocuments() {
      if (ivjAddDocuments == null) {
936      try {
           ivjAddDocuments = new javax.swing.JMenuItem();
938        ivjAddDocuments.setName("AddDocuments");
```

```
            ivjAddDocuments.setText("Add␣Documents...");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAddDocuments;
}
/**
 * Return the AddFiles1 property value.
 * @return clientapp.AddFiles
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private AddFiles getAddFiles1() {
    if (ivjAddFiles1 == null) {
        try {
            ivjAddFiles1 = new clientapp.AddFiles();
            ivjAddFiles1.setName("AddFiles1");
            ivjAddFiles1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin {1}
            ivjAddFiles1.setClientApplication(this);
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjAddFiles1;
}
/**
 * Return the ClientApplicationJMenuBar property value.
 * @return javax.swing.JMenuBar
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuBar getClientApplicationJMenuBar() {
    if (ivjClientApplicationJMenuBar == null) {
        try {
            ivjClientApplicationJMenuBar = new javax.swing.JMenuBar();
            ivjClientApplicationJMenuBar.setName("ClientApplicationJMenuBar");
            ivjClientApplicationJMenuBar.add(getFileMenu());
            ivjClientApplicationJMenuBar.add(getSetup());
            ivjClientApplicationJMenuBar.add(getHelpMenu());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjClientApplicationJMenuBar;
}
/**
 * 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)
 * @return clientapp.ClientManager
 */
public ClientManager getClientManager()
{
    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
         */
```

```
1068  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JTree getDiscussionTree() {
1070     if (ivjDiscussionTree == null) {
            try {
1072           ivjDiscussionTree = new javax.swing.JTree();
              ivjDiscussionTree.setName("DiscussionTree");
1074          ivjDiscussionTree.setToolTipText("Discussion␣tree");
              ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
1076          ivjDiscussionTree.setShowsRootHandles(true);
              ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 1));
1078          ivjDiscussionTree.setMinimumSize(new java.awt.Dimension(1, 1));
              // user code begin {1}
1080          DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode();
              ivjDiscussionTree.setModel( new DefaultTreeModel( rootNode ));
1082          ivjDiscussionTree.setRootVisible( true );
      //      ivjDiscussionTree.setCellRenderer( new DiscussionTreeCellRenderer() );
1084          ivjDiscussionTree.addMouseListener( new TopicTreeMouseAdapter( ));
              ivjDiscussionTree.addTreeSelectionListener( new TopicTreeSelectionListener() );
1086          ivjDiscussionTree.getSelectionModel().setSelectionMode( TreeSelectionModel.SINGLE_TREE_SELECTION
                    );
              ivjDiscussionTree.setShowsRootHandles( true );
1088          // make sure that the root is expanded
              ivjDiscussionTree.expandRow(0);
1090          // user code end
            } catch (java.lang.Throwable ivjExc) {
1092          // user code begin {2}
              // user code end
1094          handleException(ivjExc);
            }
1096     }
        return ivjDiscussionTree;
1098  }
      /**
1100   * Return the DocumentBox property value.
       * @return javax.swing.JComboBox
1102   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1104  private javax.swing.JComboBox getDocumentBox() {
        if (ivjDocumentBox == null) {
1106       try {
              ivjDocumentBox = new ZComboBox();
1108          ivjDocumentBox.setName("DocumentBox");
              ivjDocumentBox.setToolTipText("Visible␣documents");
1110          // user code begin {1}
              ivjDocumentBox.setEditable( false );
1112          ivjDocumentBox.addItemListener( new VisibleDocumentsBoxItemListener() );
              ivjDocumentBox.setRenderer( new VisibleDocumentsBoxCellRenderer() );
1114          ivjDocumentBox.setMaximumRowCount( 15 );
              ivjDocumentBox.insertItemAt( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION()
                    ), 0 );
1116          ivjDocumentBox.setSelectedIndex( 0 );
              ivjDocumentBox.setBorder( BorderFactory.createEtchedBorder() );
1118          // user code end
            } catch (java.lang.Throwable ivjExc) {
1120          // user code begin {2}
              // user code end
1122          handleException(ivjExc);
            }
1124     }
        return ivjDocumentBox;
1126  }
      /**
1128   * Return the Exit property value.
       * @return javax.swing.JMenuItem
1130   */
      /* 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     }
```

```
      }
1264  return ivjHelpDialog1;
    }
1266 /**
     * Return the HelpMenu property value.
1268 * @return javax.swing.JMenu
     */
1270 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenu getHelpMenu() {
1272   if (ivjHelpMenu == null) {
         try {
1274         ivjHelpMenu = new javax.swing.JMenu();
             ivjHelpMenu.setName("HelpMenu");
1276         ivjHelpMenu.setText("Help");
             ivjHelpMenu.add(getHelpTopics());
1278         ivjHelpMenu.add(getAbout());
             // user code begin {1}
1280         // help system setup
             HelpSet hs;
1282         try
             {
1284            String helpPathAndName = ClientConstants.getCLIENT_DOCUMENTATION_HELPSET_NAME();
               java.net.URL hsURL = HelpSet.findHelpSet( null, helpPathAndName );
1286           hs = new HelpSet( null, hsURL );
               HelpBroker hb = hs.createHelpBroker();
1288           getHelpTopics().addActionListener( new HelpActionListener( hb ));
             }
1290         catch ( Exception E )
             {
1292            System.err.println( ClientConstants.getCOULD_NOT_INITIALIZE_HELP() );
             }
1294         // user code end
         } catch (java.lang.Throwable ivjExc) {
1296         // user code begin {2}
             // user code end
1298         handleException(ivjExc);
         }
1300   }
      return ivjHelpMenu;
1302 }
    /**
1304  * Return the HelpTopics property value.
     * @return javax.swing.JMenuItem
1306  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
1308 private javax.swing.JMenuItem getHelpTopics() {
      if (ivjHelpTopics == null) {
1310     try {
           ivjHelpTopics = new javax.swing.JMenuItem();
1312       ivjHelpTopics.setName("HelpTopics");
           ivjHelpTopics.setText("Help␣Topics");
1314       // user code begin {1}
           // user code end
1316     } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
1318       // user code end
           handleException(ivjExc);
1320     }
      }
1322  return ivjHelpTopics;
    }
1324 /**
     * Return the JSplitPane11 property value.
1326 * @return javax.swing.JSplitPane
     */
1328 /* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
   private javax.swing.JSplitPane getInnerSplitPane() {
1330   if (ivjInnerSplitPane == null) {
         try {
1332        ivjInnerSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.HORIZONTAL_SPLIT);
            ivjInnerSplitPane.setName("InnerSplitPane");
1334        ivjInnerSplitPane.setDividerSize(6);
            ivjInnerSplitPane.setMaximumSize(new java.awt.Dimension(100000, 100000));
1336        ivjInnerSplitPane.setDividerLocation(300);
            ivjInnerSplitPane.setPreferredSize(new java.awt.Dimension(1, 1));
1338        ivjInnerSplitPane.setOneTouchExpandable(true);
            ivjInnerSplitPane.setBounds(224, 808, 798, 503);
1340        ivjInnerSplitPane.setMinimumSize(new java.awt.Dimension(1, 1));
            getInnerSplitPane().add(getJPanel11(), "left");
1342        getInnerSplitPane().add(getJPanel21(), "right");
            // user code begin {1}
1344        ivjInnerSplitPane.resetToPreferredSizes();
            // user code end
1346     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1348        // user code end
            handleException(ivjExc);
1350     }
      }
1352  return ivjInnerSplitPane;
   }
1354  /**
    * Return the JAppletContentPane property value.
1356  * @return javax.swing.JPanel
    */
1358  /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJAppletContentPane() {
1360   if (ivjJAppletContentPane == null) {
         try {
1362        ivjJAppletContentPane = new javax.swing.JPanel();
            ivjJAppletContentPane.setName("JAppletContentPane");
1364        ivjJAppletContentPane.setLayout(new java.awt.BorderLayout());
            getJAppletContentPane().add(getMessagePanel(), "South");
1366        getJAppletContentPane().add(getJToolBar1(), "North");
            getJAppletContentPane().add(getOuterSplitPane(), "Center");
1368        // user code begin {1}
            // user code end
1370     } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
1372        // user code end
            handleException(ivjExc);
1374     }
      }
1376  return ivjJAppletContentPane;
   }
1378  /**
    * Return the JPanel11 property value.
1380  * @return javax.swing.JPanel
    */
1382  /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJPanel11() {
1384   if (ivjJPanel11 == null) {
         try {
1386        ivjJPanel11 = new javax.swing.JPanel();
            ivjJPanel11.setName("JPanel11");
1388        ivjJPanel11.setPreferredSize(new java.awt.Dimension(150, 1));
            ivjJPanel11.setLayout(new java.awt.GridBagLayout());
1390        ivjJPanel11.setBackground(java.awt.SystemColor.window);
            ivjJPanel11.setMinimumSize(new java.awt.Dimension(1, 1));
1392
            java.awt.GridBagConstraints constraintsDiscussionTree = new java.awt.GridBagConstraints();
1394        constraintsDiscussionTree.gridx = 0; constraintsDiscussionTree.gridy = 0;
```

```
            constraintsDiscussionTree.fill = java.awt.GridBagConstraints.BOTH;
            constraintsDiscussionTree.anchor = java.awt.GridBagConstraints.WEST;
            constraintsDiscussionTree.weightx = 1.0;
            constraintsDiscussionTree.weighty = 1.0;
            getJPanel11().add(getDiscussionTree(), constraintsDiscussionTree);

            java.awt.GridBagConstraints constraintsJToolBar2 = new java.awt.GridBagConstraints();
            constraintsJToolBar2.gridx = 1; constraintsJToolBar2.gridy = 0;
            constraintsJToolBar2.fill = java.awt.GridBagConstraints.VERTICAL;
            constraintsJToolBar2.anchor = java.awt.GridBagConstraints.NORTHEAST;
            getJPanel11().add(getJToolBar2(), constraintsJToolBar2);
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjJPanel11;
   }
   /**
    * Return the JPanel21 property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJPanel21() {
      if (ivjJPanel21 == null) {
         try {
            ivjJPanel21 = new javax.swing.JPanel();
            ivjJPanel21.setName("JPanel21");
            ivjJPanel21.setPreferredSize(new java.awt.Dimension(450, 1));
            ivjJPanel21.setLayout(new java.awt.BorderLayout());
            ivjJPanel21.setMinimumSize(new java.awt.Dimension(1, 1));
            getJPanel21().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 ivjJPanel21;
   }
   /**
    * Return the JPanel3 property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getJPanel3() {
      if (ivjJPanel3 == null) {
         try {
            ivjJPanel3 = new javax.swing.JPanel();
            ivjJPanel3.setName("JPanel3");
            ivjJPanel3.setLayout(new java.awt.GridBagLayout());

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

            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        }
```

```
      }
1726  return ivjJSeparator6;
    }
1728 /**
     * Return the JSeparator7 property value.
1730 * @return javax.swing.JSeparator
     */
1732 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JSeparator getJSeparator7() {
1734  if (ivjJSeparator7 == null) {
        try {
1736      ivjJSeparator7 = new javax.swing.JSeparator();
          ivjJSeparator7.setName("JSeparator7");
1738      // user code begin {1}
          // user code end
1740    } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
1742      // user code end
          handleException(ivjExc);
1744    }
      }
1746  return ivjJSeparator7;
    }
1748 /**
     * Return the JSeparator9 property value.
1750 * @return javax.swing.JSeparator
     */
1752 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JSeparator getJSeparator9() {
1754  if (ivjJSeparator9 == null) {
        try {
1756      ivjJSeparator9 = new javax.swing.JSeparator();
          ivjJSeparator9.setName("JSeparator9");
1758      // user code begin {1}
          // user code end
1760    } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
1762      // user code end
          handleException(ivjExc);
1764    }
      }
1766  return ivjJSeparator9;
    }
1768 /**
     * Return the JToolBar1 property value.
1770 * @return javax.swing.JToolBar
     */
1772 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JToolBar getJToolBar1() {
1774  if (ivjJToolBar1 == null) {
        try {
1776      ivjJToolBar1 = new javax.swing.JToolBar();
          ivjJToolBar1.setName("JToolBar1");
1778      ivjJToolBar1.setFloatable(false);
          getJToolBar1().add(getJPanel3(), getJPanel3().getName());
1780      // user code begin {1}
          // user code end
1782    } catch (java.lang.Throwable ivjExc) {
          // user code begin {2}
1784      // user code end
          handleException(ivjExc);
1786    }
      }
1788  return ivjJToolBar1;
    }
1790 /**
```

```
     * 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
1808       } 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
1834       } 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");
            ivjPanoramaPanel.setToolTipText("null");
            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);
         }
      }
      return ivjPanoramaPanel;
   }
   /**
    * Return the JToolBarButton4 property value.
    * @return javax.swing.JButton
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JButton getPreferencesButton() {
      if (ivjPreferencesButton == null) {
         try {
            ivjPreferencesButton = new javax.swing.JButton();
            ivjPreferencesButton.setName("PreferencesButton");
            ivjPreferencesButton.setText("");
            ivjPreferencesButton.setToolTipText( "Preferences" );
            ivjPreferencesButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjPreferencesButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
            ivjPreferencesButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                toolbarButtonGraphics/development/Host24.gif")));
            ivjPreferencesButton.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 ivjPreferencesButton;
   }
   /**
    * Return the PreferencesDialog1 property value.
    * @return clientapp.PreferencesDialog
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private PreferencesDialog getPreferencesDialog1() {
      if (ivjPreferencesDialog1 == null) {
         try {
            ivjPreferencesDialog1 = new clientapp.PreferencesDialog();
            ivjPreferencesDialog1.setName("PreferencesDialog1");
            ivjPreferencesDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
            // user code begin {1}
            ivjPreferencesDialog1.setClientApplication( this );
            ivjPreferencesDialog1.init();
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjPreferencesDialog1;
   }
   /**
```

```
2178    * Return the OptionsItem property value.
        * @return javax.swing.JMenuItem
2180    */
       /* WARNING: THIS METHOD WILL BE REGENERATED. */
2182   private javax.swing.JMenuItem getPreferencesItem() {
          if (ivjPreferencesItem == null) {
2184          try {
                 ivjPreferencesItem = new javax.swing.JMenuItem();
2186             ivjPreferencesItem.setName("PreferencesItem");
                 ivjPreferencesItem.setText("Preferences");
2188             // user code begin {1}
                 // user code end
2190          } catch (java.lang.Throwable ivjExc) {
                 // user code begin {2}
2192             // user code end
                 handleException(ivjExc);
2194          }
          }
2196      return ivjPreferencesItem;
       }
2198   /**
        * Return the RemoveDocuments property value.
2200    * @return javax.swing.JMenuItem
        */
2202   /* WARNING: THIS METHOD WILL BE REGENERATED. */
       public javax.swing.JMenuItem getRemoveDocuments() {
2204      if (ivjRemoveDocuments == null) {
             try {
2206             ivjRemoveDocuments = new javax.swing.JMenuItem();
                 ivjRemoveDocuments.setName("RemoveDocuments");
2208             ivjRemoveDocuments.setText("Remove␣Documents...");
                 // user code begin {1}
2210             // user code end
             } catch (java.lang.Throwable ivjExc) {
2212             // user code begin {2}
                 // user code end
2214             handleException(ivjExc);
             }
2216      }
          return ivjRemoveDocuments;
2218   }
       /**
2220    * Return the RemoveFiles1 property value.
        * @return clientapp.RemoveFiles
2222    */
       /* WARNING: THIS METHOD WILL BE REGENERATED. */
2224   private RemoveFiles getRemoveFiles1() {
          if (ivjRemoveFiles1 == null) {
2226           try {
                 ivjRemoveFiles1 = new clientapp.RemoveFiles();
2228             ivjRemoveFiles1.setName("RemoveFiles1");
                 // user code begin {1}
2230             ivjRemoveFiles1.setClientApplication( this );
                 ivjRemoveFiles1.initialize();
2232             // user code end
             } catch (java.lang.Throwable ivjExc) {
2234             // user code begin {2}
                 // user code end
2236             handleException(ivjExc);
             }
2238      }
          return ivjRemoveFiles1;
2240   }
       /**
2242    * 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}
2504     // user code end
      } catch (java.lang.Throwable ivjExc) {
2506     // user code begin {3}
         // user code end
2508     handleException(ivjExc);
      }
2510  }
      /**
2512   * connEtoM6: (PreferencesItem.action.actionPerformed(java.awt.event.ActionEvent) --> PreferencesDialog1
           .show()V)
       * @param arg1 java.awt.event.ActionEvent
2514   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
2516  private void PreferencesMenuItemToDialog(java.awt.event.ActionEvent arg1) {
      try {
2518     // user code begin {1}
         // user code end
2520     getPreferencesDialog1().show();
         // user code begin {2}
2522     // user code end
      } catch (java.lang.Throwable ivjExc) {
2524     // user code begin {3}
         // user code end
2526     handleException(ivjExc);
      }
2528  }
      /**
2530   * Prepare termination of the object.
       * Creation date: (28-07-00 00:34:11)
2532   */
      public void prepareTermination()
2534  {
      if ( getClientManager().getClientInfo().isLocalMode() == false )
2536  {
         getClientManager().disconnectFromGroup( getClientManager().getClientInfo().getActiveGroup() );
2538  }
      getClientManager().informServerOfExit();
2540  getClientManager().getClientInfo().writePreferences();
      setTerminated( true );
2542  }
      /**
2544   * Remove a document from the visible document list box.
       * Creation date: (30-08-00 15:43:31)
2546   * @param senderName java.lang.String
       * @param documentName java.lang.String
2548   */
      public void removeDocumentFromVisibleList(String senderName, String documentName)
2550  {
      String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
2552  if ( getVisibleDocumentsList().containsKey( senderName ))
      {
2554     ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
         for ( int i = 0; i < docList.size(); i++ )
2556     {
            if ( docList.get( i )instanceof DocumentItem )
2558        {
               if ( ( (DocumentItem) docList.get( i )).getDocumentID().equals( fqPath ))
2560           {
                  docList.remove( i );
2562           }
            }
2564     }
         // if group member no longer contributes to panorama, i.e. if the author's name is the only entry
              ,then remove the list
2566     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;
26   private peerviewmisc.Message currentMessage = null;
   private java.util.HashSet localFiles = new HashSet();
28   private java.awt.datatransfer.Clipboard clipBoard = new Clipboard( ClientConstants.getCLIPBOARD_NAME
       () );
   private boolean localMode = true;
30 /**
 * ClientInfo constructor comment.
32 */
public ClientInfo() {
34   super();
   initialize();
36 }
/**
38 * 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.
40 * Creation date: (16-07-00 13:50:53)
 * @return java.lang.String
42 */
public static String createUniqueClientID()
44 {
   String ID = ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )+ (new java.util.Date
       ().toString() );
46   try
   {
48   ID = ID + java.net.InetAddress.getLocalHost().toString();
   }
50   catch ( java.net.UnknownHostException UHE )
   {
52      System.err.println( UHE.toString() );
      System.exit( 0 );
54   }
   return ID;
56 }
/**
58 * Insert the method's description here.
 * Creation date: (16-07-00 13:38:00)
60 * @exception java.lang.Throwable The exception description.
 */
62 protected void finalize() throws java.lang.Throwable
{
64   // write preferences
}
66 /**
 * Insert the method's description here.
68 * Creation date: (06-07-00 13:15:33)
 * @return peerviewmisc.WorkGroup
70 */
public peerviewmisc.WorkGroup getActiveGroup() {
72   return activeGroup;
}
74 /**
 * Insert the method's description here.
76 * Creation date: (07-08-00 15:57:43)
 * @return java.awt.datatransfer.Clipboard
78 */
public java.awt.datatransfer.Clipboard getClipBoard() {
80   return clipBoard;
}
82 /**
 * Insert the method's description here.
84 * Creation date: (11-07-00 15:18:55)
 * @return java.lang.String
86 */
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.
```

```
152   * Creation date: (16-07-00 13:36:57)
      */
154   public void initialize()
      {
156   }
      /**
158    * Insert the method's description here.
       * Creation date: (10-08-00 16:37:24)
160    * @return boolean
       */
162   public boolean isLocalMode() {
        return localMode;
164   }
      /**
166    * Obligatory method for meeting the ClipboardOwner interface specification
       * Creation date: (07-08-00 17:28:28)
168    * @param clipBoard java.awt.datatransfer.Clipboard
       * @param contents java.awt.datatransfer.Transferable
170    */
      public void lostOwnership(Clipboard clipBoard, Transferable contents)
172   {
      }
174   /**
       * Read preferences from Properties object on disk. Each call to this function results in a read
176    * 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
178    * plain text.
       * Algorithm:
180    * 1. Construct new Properties object P
       * 2. Initialize P with default preferences
182    * 3. Read user preferences from disk file
       * 4. IF read is successful THEN
184    * 4.1. Store user preferences in P
       * 5. Return P
186    * This approach will return a valid object even if the read fails or in case of the application being
           used for the first
       * time.
188    * Creation date: (25-06-00 13:54:55)
       */
190   public static void readPreferences()
      {
192     try
        {
194       FileInputStream f = new FileInputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
          preferences.load(f);
196       f.close();
        }
198     catch (FileNotFoundException ffe)
        {
200       // the exception can be handled by ignoring it since the default values in p will substitute for
              what should
          // have been read from file.
202     }
        catch (IOException ie)
204     {
          // any IOException != FileNotFound should be dealt with in the conventional manner
206       System.err.println( ie.toString() );
          System.exit(0);
208     }
      }
210   /**
       * Insert the method's description here.
212    * 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) {
```

```
280     localMode = newLocalMode;
    }
282 /**
     * Insert the method's description here.
284  * Creation date: (02-07-00 22:08:51)
     * @param newPreferences java.util.Properties
286  */
    public static void setPreferences(java.util.Properties newPreferences) {
288     preferences = newPreferences;
    }
290 /**
     * Write preferences to disk file.
292  * Creation date: (17-07-00 12:13:32)
     * @param preferences java.util.Properties
294  */
    public static void writePreferences()
296 {
       // write Properties object to disk
298    try
       {
300       FileOutputStream f = new FileOutputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
          preferences.store( f, ClientConstants.getPROPERTIES_HEADER() );
302       f.close();
       }
304    catch (IOException ie)
       {
306       System.err.println( ie.toString() );
          System.exit(0);
308    }

310 }
    }
```

Listing C.5: ClientManager.java

```
    package clientapp;
 2

 4  import java.io.*;
    import javax.swing.JComponent;
 6  import javax.swing.*;
    import javax.swing.ImageIcon;
 8  import javax.swing.JEditorPane;
    import javax.swing.JTextArea;
10  import com.sun.media.jsdt.*;
    import com.sun.media.jsdt.event.*;
12  import java.util.Properties;
    import java.util.Hashtable;
14  import peerviewmisc.*;
    import java.util.zip.*;
16  import java.util.Enumeration;
    import java.util.Iterator;
18  import java.util.Random;
    import javax.swing.tree.*;
20  import javax.swing.event.*;
    import java.util.GregorianCalendar;
22  import java.awt.datatransfer.*;

24  /**
     * The main control class for the client application. This class holds methods and control structures
         shared by several
26   * client application components.
     * Creation date: (14-06-00 21:27:43)
28   * @author:
     */
30  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();
336         String authorName = incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() );
            owner.initMessageArea( 1, documents.size(), 1, ClientConstants.getADDING_DISTRIBUTED_DOCUMENTS(
                authorName, documents.size() ));
338         for ( int i = 0 ; i < documents.size(); i++ )
            {
340            CompressedDocumentPackage compressedDocumentPackage = (CompressedDocumentPackage) documents.
                   get( i );
               byte[] rawDocumentData = inflateDocument( compressedDocumentPackage );
342            owner.getPanoramaPanel().addGlobalDocument( rawDocumentData, senderName, authorName,
                   compressedDocumentPackage.getPath(), compressedDocumentPackage.getName() );
               owner.advanceTask();
344            yield();
            }
346         updatePanorama();
            owner.resetMessageArea();
348         owner.lockInterface( false );
         }
350   };

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

358      public GroupManagementClient()
         {
360      }

362      public String getName()
         {
364         return name;
         }
366
         public void setName( String nameArg )
368      {
            name = nameArg;
370      }

372      /** Accept and parse incoming data. The if-else structure maps message types to functions.
         */
374      public synchronized void dataReceived( Data data )
         {
376         DataPackage incomingPackage = null;
            try
378         {
               incomingPackage = (DataPackage) data.getDataAsObject();
380         }
            catch ( Exception E )
382         {
               System.err.println( E.toString() );
384            System.exit(0);
            }
386         if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase ( ClientConstants.getGROUP_DIRECTORY() ))
            {
388            updateGroupDirectory( (Hashtable) incomingPackage.getContents() );
            }
390         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
                equalsIgnoreCase ( ClientConstants.getNEW_DOCUMENT() ))
            {
392            addDistributedDocument( incomingPackage, data.getSenderName() );
            }
394         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
    */
674public 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;
```

```
    }
1052 /**
     * Inform other members of the group that the client belongs to that the document designated by the
         parameter should
1054  be removed from their panoramas.
     * Creation date: (15-07-00 15:46:56)
1056  * @param fullyQualifiedDocumentName java.lang.String
     */
1058 public void informGroupMembersOfDocumentRemoval(String fullyQualifiedDocumentName)
     {
1060    DataPackage outgoingPackage = new DataPackage();

1062    outgoingPackage.setContents( fullyQualifiedDocumentName );
       outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getCLIENT_REMOVED_LOCAL_DOCUMENTS() );
1064    try
       {
1066      if ( getClientInfo().isLocalMode() == false )
          {
1068        URLString url = clientInfo.getActiveGroup().getSessionURL();
          Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
1070        Channel channel = session.createChannel( getGroupManagementClient(), clientInfo.getActiveGroup()
                .getWorkGroupChannelID(), true, true, false );
          channel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));
1072      }
       }
1074    catch ( JSDTException JSDTE )
       {
1076      handleJSDTException( JSDTE );
       }
1078 }
     /**
1080  * Inform other group members that the panorama has changed and must be updated.
     * Creation date: (31-08-00 19:11:21)
1082  */
     public void informGroupMembersOfPanoramaUpdate()
1084 {
       if ( getClientInfo().isLocalMode() == false )
1086   {
          URLString url;
1088      Session session;
          Channel groupChannel = null;
1090        try
          {
1092        url = getClientInfo().getActiveGroup().getSessionURL();
          session = SessionFactory.createSession( getGroupManagementClient(), url, false );
1094         groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
                getActiveGroup().getWorkGroupChannelID(), true, true, false);

1096        DataPackage outgoingPackage = new DataPackage();
          outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
               getUPDATE_PANORAMA() );
1098      groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ));

1100      }
          catch ( JSDTException JSDTE )
1102      {
            handleJSDTException( JSDTE );
1104      }
       }
1106 }
     /**
1108  * Notify the server that this client has terminated.
     * Creation date: (01-11-00 01:16:44)
1110  */
     public void informServerOfExit()
```

```
{
   DataPackage outgoingPackage = new DataPackage();

   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
        getCLIENT_LEFT() );
   try
   {
      if ( getClientInfo().getGroupManagementChannel() != null )
      {
         getClientInfo().getGroupManagementChannel().sendToClient(
            getGroupManagementClient(),
            ClientInfo.getPreferences().getProperty( ClientConstants.getSERVER_NAME() ),
            new Data( outgoingPackage ));
      }
   }
   catch ( JSDTException JSDTE )
   {
      handleJSDTException( JSDTE );
   }
}
/**
 * Inform the server that the client has exited the group to which it belonged, if any.
 * Creation date: (31-10-00 23:55:01)
 */
public void informServerOfGroupExit()
{
   if ( getClientInfo().getActiveGroup() != null )
   {
      DataPackage outgoingPackage = new DataPackage();

      outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
           getCLIENT_LEFT_GROUP() );
      outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
           getActiveGroup().getWorkGroupID() );
      try
      {
         if ( getClientInfo().getGroupManagementChannel() != null )
         {
            getClientInfo().getGroupManagementChannel().sendToClient(
               getGroupManagementClient(),
               ClientInfo.getPreferences().getProperty( ClientConstants.getSERVER_NAME() ),
               new Data( outgoingPackage ));
         }
      }
      catch ( JSDTException JSDTE )
      {
         handleJSDTException( JSDTE );
      }
   }
}
/**
 * Inform the server that the client has added one or more documents to the panorama.
 * Creation date: (04-10-00 11:39:50)
 * @param docPath java.lang.String
 */
public void informServerOfNewDocument(String docPath)
{
   DataPackage outgoingPackage = new DataPackage();

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

```
         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;
```

```
    }
1608 /**
     * Insert the method's description here.
1610 * Creation date: (09-07-00 22:48:06)
     * @param newTree javax.swing.tree.DefaultTreeModel
1612 * @param documentName java.lang.String
     */
1614 public void setTopicTree(DefaultTreeModel newTree, String documentName)
     {
1616    setDisplayedMessage( null );
       getClientInfo().setCurrentDocumentName( documentName );
1618    owner.setTopicTree( newTree );
     }
1620 /**
     * Update one of the distributed documents shown in the panorama.
1622 * Creation date: (21-08-00 17:30:56)
     * @param documentName java.lang.String
1624 * @param senderName java.lang.String
     * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
1626 */
     public void updateDistributedDocument(String documentName, String senderName, CompressedDocumentPackage
         compressedDifferenceBitmapPackage)
1628 {
       owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATING_DISTRIBUTED_DOCUMENT( documentName ));
1630   byte rawFileContents[] = inflateDocument( compressedDifferenceBitmapPackage );
       owner.getPanoramaPanel().updateDistributedDocument( senderName, documentName, rawFileContents );
1632 }
     /**
1634 * 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.
1636 * Creation date: (21-08-00 16:01:13)
     * @param path java.lang.String
1638 * @param clientName java.lang.String
     * @param oldRawFileContents byte[]
1640 * @param newRawFileContents byte[]
     */
1642 public void updateDocument(String path, String clientName, byte[] oldRawFileContents, byte[]
         newRawFileContents)
     {
1644    byte differenceBitmap[] = computeBitmapCombination( oldRawFileContents, newRawFileContents );
       CompressedDocumentPackage compressedDifferenceBitmapPackage = deflateDocument( differenceBitmap );
1646   broadcastDocumentUpdate( path, compressedDifferenceBitmapPackage );
     }
1648 /**
     * Insert the method's description here.
1650 * Creation date: (04-08-00 15:50:41)
     * @param newDirectory java.util.Hashtable
1652 */
     public void updateGroupDirectory(Hashtable newDirectory)
1654 {
       getClientInfo().setGroups( newDirectory );
1656   owner.getGroupDirectory1().updateTable();
       if ( getClientInfo().getActiveGroup() != null )
1658   {
          getClientInfo().setActiveGroup( (WorkGroup) newDirectory.get( getClientInfo().getActiveGroup().
              getWorkGroupID() ));
1660   }
     }
1662 /**
     * Insert the method's description here.
1664 * Creation date: (14-06-00 23:07:20)
     */
1666 public void updatePanorama()
     {
1668   owner.updateVisibleDocumentsBox();
```

```
      // System.out.println("updatePanorama");
1670  owner.getPanoramaPanel().validateLayout();
    }
1672  /**
     * Insert the method's description here.
1674  * Creation date: (31-08-00 02:14:49)
     */
1676  public void updatePanoramaByRequest()
    {
1678  owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() );
      updatePanorama();
1680  }
    /**
1682  * Insert the method's description here.
     * Creation date: (27-08-00 21:21:06)
1684  * @param fullyQualifiedDocumentName java.lang.String
     * @param treeUpdate javax.swing.tree.DefaultTreeModel
1686  */
    public void updateTopicTree(String senderName, String documentName, DefaultTreeModel treeUpdate)
1688  {
      String fqName = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
1690  if ( getClientInfo().getCurrentDocumentName().equals( fqName ))
      {
1692    setTopicTree( treeUpdate, fqName );
      }
1694  }
    /**
1696  * Test connection to group management and notify user if it isn't active.
     * Creation date: (09-10-00 18:57:46)
1698  * @return boolean
     */
1700  public boolean verifyGroupConnectionManagement()
    {
1702  if ( getClientInfo().getGroupManagementSession() == null )
      {
1704    owner.initMessageArea( 1, 1, 1, ClientConstants.getNOT_CONNECTED_TO_SERVER() );
      return false;
1706  }
      else
1708  {
      return true;
1710  }
    }
1712  /**
     * Extract a list of documents displayed in the panorama and write it to disk. This can then be restored
         automatically
1714  * the next time the client is started.
     * Creation date: (06-09-00 21:48:43)
1716  */
    public void writeDocumentList()
1718  {
      try
1720  {
      FileOutputStream fos = new FileOutputStream( ClientConstants.getDOCUMENT_LIST_FILENAME() );
1722  ObjectOutputStream oos = new ObjectOutputStream( fos );

1724  java.util.Vector docs = new java.util.Vector();
      Iterator iterator = owner.getPanoramaPanel().getLocalDocuments().values().iterator();
1726  while ( iterator.hasNext() )
      {
1728    edu.umd.cs.jazz.ZVisualLeaf leaf = (edu.umd.cs.jazz.ZVisualLeaf) iterator.next();
      DocPackage docPackage = new DocPackage( (String) leaf.getClientProperty( ClientConstants.
          getDOCUMENT_NAME() ),
1730                              new java.awt.Dimension( (int) leaf.getVisualComponent().getBounds().
                                      getMaxX(),
```

```
                                                    (int) leaf.getVisualComponent().getBounds().getMaxY
                                                        () ),
1732                                    new java.awt.Dimension( (int) leaf.editor().getTransformGroup().
                                                        getTranslateX(),
                                                    (int) leaf.editor().getTransformGroup().
                                                        getTranslateY() )
1734                                    );
            docPackage.setNodeLocked( (String) leaf.getClientProperty( ClientConstants.getNODE_LOCKED() ));
1736        docs.add( docPackage );
          }
1738      oos.writeObject( docs );
          oos.flush();
1740      oos.close();
        }
1742    catch ( IOException ioe )
        {
1744      System.err.println( ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
          owner.initMessageArea( 1, 1, 1, ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
1746    }

1748  }
      }
```

Listing C.6: ClientConstants.java

```
      package clientapp;
   2

   4   import java.awt.Color;
       import java.util.Hashtable;
   6   import edu.umd.cs.jazz.*;
       import edu.umd.cs.jazz.component.*;
   8   import edu.umd.cs.jazz.event.*;
       import edu.umd.cs.jazz.io.*;
  10   import edu.umd.cs.jazz.util.*;
       import java.util.Properties;
  12   import peerviewmisc.SharedConstants;
       import java.util.Random;
  14
       /**
  16   * 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)
  18   * @author:
       */
  20   public class ClientConstants extends SharedConstants {

  22     private final static Color BACKGROUND_COLOUR = new Color(200, 200, 200);
         public final static java.lang.String PROCESSING_DOCUMENTS = "Adding␣documents␣to␣the␣panorama";
  24     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;
  26     private final static java.lang.String RETRIEVING_DOCUMENTS = "Adding␣documents␣to␣panorama";

  28     private final static java.lang.String PLAIN_TEXT = "text/plain";
         private final static java.lang.String HTML = "text/html";
  30     private final static java.lang.String RTF = "text/rtf";
         private final static java.lang.String BITMAP = "bitmap";
  32     private final static java.lang.String ALL_FILES = "All␣files";
         private final static java.lang.String[][] FileTypes = {  {".txt", "Plain␣text␣(.txt)", PLAIN_TEXT},
  34                                                 {".html", "HTML␣(.html,␣.htm)", HTML},
                                                     {".rtf", "RTF␣(.rtf)", RTF},
  36                                                 {".cpp", "C++␣source␣(.cpp,␣.cc)", PLAIN_TEXT},
                                                     {".h", "C++␣header␣(.h)", PLAIN_TEXT},
  38                                                 {".java", "Java␣source␣(.java)", PLAIN_TEXT},
                                                     {".gif", "GIF␣(.gif)", BITMAP},
  40                                                 {".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;
}
/**
 * Insert the method's description here.
 * Creation date: (09-08-00 23:05:17)
 * @return java.lang.String
 */
public final static java.lang.String getACTIVE_GROUP_UPON_EXIT() {
    return ACTIVE_GROUP_UPON_EXIT;
}
/**
 * Insert the method's description here.
 * Creation date: (19-08-00 20:59:44)
 * @param documentName java.lang.String
 * @param senderName java.lang.String
 */
public final static String getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(String documentName, String senderName)
{
    return "Received␣" + documentName + "␣from␣" + senderName;
}
/**
 * Insert the method's description here.
 * Creation date: (05-09-00 14:50:29)
 * @return java.lang.String
 * @param authorName java.lang.String
 * @param numDoks int
 */
public final static String getADDING_DISTRIBUTED_DOCUMENTS(String authorName, int numDoks)
{
    return "Adding␣" + numDoks + "␣documents␣received␣from␣" + authorName;
}
/**
 * Insert the method's description here.
 * Creation date: (05-10-00 23:05:28)
 * @return java.lang.String
 */
public final static java.lang.String getALL_FILES() {
    return ALL_FILES;
}
/**
 * Insert the method's description here.
 * Creation date: (25-07-00 15:52:24)
 * @return java.lang.String
 */
public final static java.lang.String getALREADY_JOINED_TO_GROUP_MESSAGE() {
    return ALREADY_JOINED_TO_GROUP_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (13-07-00 13:08:57)
 * @return java.lang.String
 */
public final static java.lang.String getANIMATION_DURATION_IN_MILLISECS() {
    return ANIMATION_DURATION_IN_MILLISECS;
}
/**
 * Insert the method's description here.
 * Creation date: (22-08-00 17:53:01)
 * @return double
 */
public final static int getANIMATION_MAX_SPEED() {
    return ANIMATION_MAX_SPEED;
}
/**
 * Insert the method's description here.
 * 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
      */
```

```
710  public final static int getDOCUMENT_BORDER_WIDTH() {
       return DOCUMENT_BORDER_WIDTH;
712  }
     /**
714   * Insert the method's description here.
      * Creation date: (22-08-00 01:26:41)
716   * @return java.lang.String
      * @param documentName java.lang.String
718   * @param authorName java.lang.String
      * @param size long
720   * @param lastUpdate java.util.Date
      */
722  public final static String getDOCUMENT_LABEL(String documentName, String authorName, long size, java.
         util.Date lastUpdate)
     {  java.util.GregorianCalendar calendar = new java.util.GregorianCalendar();
724     calendar.setTime( lastUpdate );
        return  "Name:" + documentName + "\n" +
726           "Author:␣" + authorName + "\n" +
              "Size:␣" + String.valueOf( size )+ "␣bytes" + "\n" +
728           "Last␣updated:␣" + (new java.text.SimpleDateFormat()).format( lastUpdate )+ ":" + String.valueOf
                ( calendar.get( java.util.Calendar.SECOND ));
     }
730  /**
      * Insert the method's description here.
732   * Creation date: (30-07-00 20:49:44)
      * @return java.awt.Color
734   */
     public final static java.awt.Color getDOCUMENT_LINE_BORDER_COLOUR() {
736     return DOCUMENT_LINE_BORDER_COLOUR;
     }
738  /**
      * Insert the method's description here.
740   * Creation date: (06-09-00 21:36:21)
      * @return java.lang.String
742   */
     public final static java.lang.String getDOCUMENT_LIST_FILENAME()
744  {
        return DOCUMENT_LIST_FILENAME;
746  }
     /**
748   * Insert the method's description here.
      * Creation date: (30-07-00 20:44:17)
750   * @return java.awt.Color
      */
752  public final static java.awt.Color getDOCUMENT_MATTE_BORDER_COLOUR() {
        return DOCUMENT_MATTE_BORDER_COLOUR;
754  }
     /**
756   * Insert the method's description here.
      * Creation date: (21-08-00 12:34:32)
758   * @return java.lang.String
      */
760  public final static java.lang.String getDOCUMENT_NAME() {
        return DOCUMENT_NAME;
762  }
     /**
764   * Insert the method's description here.
      * Creation date: (09-07-00 17:44:32)
766   * @return java.lang.String
      */
768  public final static java.lang.String getDOCUMENT_PATH() {
        return DOCUMENT_PATH;
770  }
     /**
772   * 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
```

```
906   */
      public final static java.lang.String getFRAME_X_COORDINATE() {
908      return FRAME_X_COORDINATE;
      }
910   /**
       * Insert the method's description here.
912    * Creation date: (03-08-00 18:09:58)
       * @return java.lang.String
914    */
      public final static java.lang.String getFRAME_Y_COORDINATE() {
916      return FRAME_Y_COORDINATE;
      }
918   /**
       * Insert the method's description here.
920    * Creation date: (25-06-00 13:32:26)
       * @return java.lang.String
922    */
      public final static java.lang.String getGRID_LAYOUT() {
924      return GRID_LAYOUT;
      }
926   /**
       * Insert the method's description here.
928    * Creation date: (19-07-00 13:00:42)
       * @return java.lang.String
930    */
      public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() {
932      return GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS;
      }
934   /**
       * Insert the method's description here.
936    * Creation date: (19-07-00 13:00:07)
       * @return java.lang.String
938    */
      public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() {
940      return GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS;
      }
942   /**
       * Insert the method's description here.
944    * Creation date: (06-09-00 15:05:25)
       * @return java.lang.String
946    */
      public final static String getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS()
948   {
         return  "Horizontal␣document␣spacing␣must␣be␣between␣" +
950            getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() + "␣and␣" +
               getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() + "." +
952            "Vertical␣document␣spacing␣must␣be␣between␣" +
               getMIN_GRID_LAYOUT_VERTICAL_SPACING() + "␣and␣" +
954            getMAX_GRID_LAYOUT_VERTICAL_SPACING() + ".";
      }
956   /**
       * Insert the method's description here.
958    * Creation date: (25-07-00 14:56:17)
       * @return java.lang.String
960    */
      public final static java.lang.String getGROUP_AT_MAXIMUM_MEMBERS_MESSAGE() {
962      return GROUP_AT_MAXIMUM_MEMBERS_MESSAGE;
      }
964   /**
       * Insert the method's description here.
966    * Creation date: (18-08-00 02:02:33)
       * @return int
968    */
      public final static String getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() {
970      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.
```

```
1302   * Creation date: (03-08-00 01:58:01)
       * @return java.lang.String
1304   */
       public final static java.lang.String getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() {
1306      return PANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE;
       }
1308   /**
       * Insert the method's description here.
1310   * Creation date: (22-06-00 20:20:00)
       * @return java.lang.String
1312   */
       public final static java.lang.String getPLAIN_TEXT() {
1314      return PLAIN_TEXT;
       }
1316   /**
       * Insert the method's description here.
1318   * Creation date: (25-06-00 19:29:14)
       * @return java.lang.String
1320   */
       public final static java.lang.String getPREFERENCES_FILE_NAME() {
1322      return PREFERENCES_FILE_NAME;
       }
1324   /**
       * Insert the method's description here.
1326   * Creation date: (17-06-00 19:40:40)
       * @return java.lang.String
1328   */
       public final static java.lang.String getPROCESSING_DOCUMENTS() {
1330      return PROCESSING_DOCUMENTS;
       }
1332   /**
       * Insert the method's description here.
1334   * Creation date: (26-06-00 08:13:08)
       * @return java.lang.String
1336   */
       public final static java.lang.String getPROPERTIES_HEADER() {
1338      return PROPERTIES_HEADER;
       }
1340   /**
       * Insert the method's description here.
1342   * Creation date: (21-08-00 15:49:38)
       * @return java.lang.String
1344   */
       public final static java.lang.String getRAW_BITMAP() {
1346      return RAW_BITMAP;
       }
1348   /**
       * Insert the method's description here.
1350   * Creation date: (19-08-00 21:35:31)
       * @param documentName java.lang.String
1352   * @param senderName java.lang.String
       */
1354   public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName, String senderName)
       {
1356      return senderName + "␣has␣requested␣that␣" + documentName + "␣be␣removed␣from␣the␣panorama.";
       }
1358   /**
       * Insert the method's description here.
1360   * Creation date: (05-09-00 15:15:12)
       * @param authorName java.lang.String
1362   * @param numDoks int
       */
1364   public final static String getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST(String authorName, int numDoks)
       {
1366      return "Removing␣" + numDoks + "␣documents␣from␣the␣panorama␣by␣request␣from␣" + authorName;
       }
```

```
1368  /**
       * Insert the method's description here.
1370   * Creation date: (19-08-00 21:32:47)
       * @return java.lang.String
1372   */
      public final static java.lang.String getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() {
1374     return REMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE;
      }
1376  /**
       * Insert the method's description here.
1378   * Creation date: (23-06-00 09:53:56)
       * @return java.lang.String
1380   */
      public final static java.lang.String getREMOVING_DOCUMENTS() {
1382     return REMOVING_DOCUMENTS;
      }
1384  /**
       * Insert the method's description here.
1386   * Creation date: (01-08-00 00:35:49)
       * @return float
1388   */
      public final static float getRENDER_CUTOFF() {
1390     return RENDER_CUTOFF;
      }
1392  /**
       * Insert the method's description here.
1394   * Creation date: (08-07-00 20:37:18)
       * @return java.lang.String
1396   */
      public final static java.lang.String getREQUEST_DOCUMENTS() {
1398     return REQUEST_DOCUMENTS;
      }
1400  /**
       * Insert the method's description here.
1402   * Creation date: (08-07-00 20:24:59)
       * @param numberOfDocuments String
1404   */
      public static String getREQUESTING_DOCUMENTS_MESSAGE(int numberOfDocuments)
1406  {
        return "Receiving␣and␣adding␣a␣total␣of␣" + String.valueOf( numberOfDocuments )+ "␣documents␣from␣the
            ␣other␣members␣of␣the␣group.";
1408  }
      /**
1410   * Insert the method's description here.
       * Creation date: (10-08-00 15:46:42)
1412   * @return java.lang.String
       */
1414  public final static java.lang.String getREQUESTING_GROUPS_DIRECTORY() {
        return REQUESTING_GROUPS_DIRECTORY;
1416  }
      /**
1418   * Insert the method's description here.
       * Creation date: (08-08-00 09:52:01)
1420   * @return int
       */
1422  public final static int getRESET_MESSAGE_AREA_DELAY() {
        return RESET_MESSAGE_AREA_DELAY;
1424  }
      /**
1426   * Insert the method's description here.
       * Creation date: (28-07-00 09:40:21)
1428   * @return java.awt.Color
       */
1430  public final static java.awt.Color getRESIZE_RECTANGLE_PEN_COLOUR() {
        return RESIZE_RECTANGLE_PEN_COLOUR;
1432  }
```

```
    /**
1434  * Insert the method's description here.
      * Creation date: (19-06-00 22:01:04)
1436  * @return java.lang.String
      */
1438  public final static java.lang.String getRETRIEVING_DOCUMENTS() {
        return RETRIEVING_DOCUMENTS;
1440  }
    /**
1442  * Insert the method's description here.
      * Creation date: (22-06-00 20:21:32)
1444  * @return java.lang.String
      */
1446  public final static java.lang.String getRTF() {
        return RTF;
1448  }
    /**
1450  * Insert the method's description here.
      * Creation date: (14-07-00 14:38:06)
1452  * @return double
      */
1454  public final static double getSCALE_FACTOR_SCALING() {
        return SCALE_FACTOR_SCALING;
1456  }
    /**
1458  * Insert the method's description here.
      * Creation date: (30-08-00 16:45:24)
1460  * @return java.lang.String
      */
1462  public final static java.lang.String getSENDER_NAME() {
        return SENDER_NAME;
1464  }
    /**
1466  * Insert the method's description here.
      * Creation date: (30-08-00 17:24:57)
1468  * @return java.lang.String
      */
1470  public final static java.lang.String getSHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX() {
        return SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX;
1472  }
    /**
1474  * Insert the method's description here.
      * Creation date: (25-06-00 16:35:18)
1476  * @return java.lang.String
      */
1478  public final static java.lang.String getSIGNATURE() {
        return SIGNATURE;
1480  }
    /**
1482  * Insert the method's description here.
      * Creation date: (05-10-00 23:11:42)
1484  * @return int
      */
1486  public final static int getTEXT_DOCUMENT_THRESHOLD() {
        return TEXT_DOCUMENT_THRESHOLD;
1488  }
    /**
1490  * Insert the method's description here.
      * Creation date: (11-08-00 00:22:49)
1492  * @return java.lang.String
      */
1494  public final static java.lang.String getTITLE_REQUIRED_MESSAGE() {
        return TITLE_REQUIRED_MESSAGE;
1496  }
    /**
1498  * 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;
```

```
1630  }
      /**
1632   * Insert the method's description here.
       * Creation date: (15-08-00 17:49:18)
1634   * @return java.lang.String
       */
1636  public final static java.lang.String getZOOM_TO_OVERVIEW() {
         return ZOOM_TO_OVERVIEW;
1638  }
      /**
1640   * Insert the method's description here.
       * Creation date: (15-08-00 17:40:30)
1642   * @return java.lang.String
       */
1644  public final static java.lang.String getZOOM_TO_OVERVIEW_ON_UPDATE() {
         return ZOOM_TO_OVERVIEW_ON_UPDATE;
1646  }
      /**
1648   * Insert the method's description here.
       * Creation date: (30-07-00 19:10:52)
1650   * @return java.awt.Color
       */
1652  public final static java.awt.Color getZOOM_VECTOR_COLOUR() {
         return ZOOM_VECTOR_COLOUR;
1654  }
      /**
1656   * Insert the method's description here.
       * Creation date: (31-07-00 18:02:18)
1658   * @return int
       */
1660  public final static int getZOOM_VECTOR_WIDTH() {
         return ZOOM_VECTOR_WIDTH;
1662  }
      /**
1664   * Insert the method's description here.
       * Creation date: (22-06-00 19:51:35)
1666   */
      public void initialize()
1668  {

1670  }
      /**
1672   * Insert the method's description here.
       * Creation date: (23-02-01 08:42:21)
1674   * @param newFILE_TYPE_SYNONYMS java.util.Hashtable
       */
1676  public static void setFILE_TYPE_SYNONYMS(java.util.Hashtable newFILE_TYPE_SYNONYMS) {
         FILE_TYPE_SYNONYMS = newFILE_TYPE_SYNONYMS;
1678  }
      }
```

Listing C.7: ClientPanorama.java

```
      package clientapp;
  2
      /**
  4    * This class represents the client panorama
       * Creation date: (15-06-00 00:26:38)
  6    * @author:
       */
  8   import java.awt.Color;
      import java.awt.Dimension;
 10   import java.awt.event.*;
      import java.lang.reflect.InvocationTargetException;
 12   import java.util.Hashtable;
      import java.io.*;
 14   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;
      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()
```

```
{
574    ZNode nodes[] = new ZNode[ getLocalDocuments().size() + getDocuments().size() ];
       int i = 0;
576    for (int j = 0 ; j < getLocalDocuments().size(); j++, i++ )
       {
578       nodes[ i ] = (ZNode) getLocalDocuments().values().toArray()[ j ];
       }
580    for ( int j = 0 ; j < getDocuments().size(); j++, i++ )
       {
582       nodes[ i ] = (ZNode) getDocuments().values().toArray()[ j ];
       }
584    return nodes;
   }
586 /**
    * Insert the method's description here.
588 * Creation date: (31-07-00 19:43:21)
    * @return edu.umd.cs.jazz.util.ZBounds
590 */
   public edu.umd.cs.jazz.util.ZBounds getBoundsBeforeResizing() {
592    return boundsBeforeResizing;
   }
594 /**
    * Return the CopyText property value.
596 * @return javax.swing.JMenuItem
    */
598 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getCopyTextItem() {
600    if (ivjCopyTextItem == null) {
          try {
602          ivjCopyTextItem = new javax.swing.JMenuItem();
             ivjCopyTextItem.setName("CopyTextItem");
604          ivjCopyTextItem.setMnemonic('C');
             ivjCopyTextItem.setText("Copy␣text");
606          // user code begin {1}
             ivjCopyTextItem.addActionListener( new PopUpActionListener() );
608          // user code end
          } catch (java.lang.Throwable ivjExc) {
610          // user code begin {2}
             // user code end
612          handleException(ivjExc);
          }
614    }
       return ivjCopyTextItem;
616 }
   /**
618 * Insert the method's description here.
    * Creation date: (28-07-00 00:50:07)
620 * @return edu.umd.cs.jazz.ZVisualLeaf
    */
622 public edu.umd.cs.jazz.ZVisualLeaf getCurrentlyClickedNode() {
       return currentlyClickedNode;
624 }
   /**
626 * Helper method to return a document node, irrespective of whether it is global or local
    * Creation date: (14-10-00 11:47:11)
628 * @return edu.umd.cs.jazz.ZVisualLeaf
    * @param clientName java.lang.String
630 * @param documentName java.lang.String
    */
632 public ZVisualLeaf getDocument(String clientName, String documentName)
   {
634    String fqName = ClientConstants.createFullyQualifiedDocumentName( clientName, documentName );
       ZVisualLeaf returnLeaf = null;
636    if ( localDocuments.get( fqName )!= null )
       {
638       returnLeaf = (ZVisualLeaf) getLocalDocuments().get( fqName );
```

```
    }
640    else
    {
642       returnLeaf = (ZVisualLeaf) getDocuments().get( fqName );
    }
644    return returnLeaf;
  }
646  /**
     * Return the PopupMenu property value.
648  * @return javax.swing.JPopupMenu
     */
650  /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private JPopupMenu getDocumentMenu() {
652    if (ivjDocumentMenu == null) {
       try {
654       ivjDocumentMenu = new JPopupMenu();
          ivjDocumentMenu.setName("DocumentMenu");
656       ivjDocumentMenu.add(getResizeItem());
          ivjDocumentMenu.add(getMoveItem());
658       ivjDocumentMenu.add(getIvjLockItem());
          ivjDocumentMenu.add(getJSeparator2());
660       ivjDocumentMenu.add(getCopyTextItem());
          ivjDocumentMenu.add(getIvjSeparator3());
662       ivjDocumentMenu.add(getIvjUnlockAllItem());
          ivjDocumentMenu.add(getIvjArrangeAllItem());
664       ivjDocumentMenu.add(getIvjAutoLockAllItem());
          // user code begin {1}
666       // user code end
       } catch (java.lang.Throwable ivjExc) {
668       // user code begin {2}
          // user code end
670       handleException(ivjExc);
       }
672    }
    return ivjDocumentMenu;
674  }
    /**
676  * Insert the method's description here.
     * Creation date: (08-07-00 01:30:08)
678  * @return java.util.Hashtable
     */
680  public java.util.Hashtable getDocuments() {
    return documents;
682  }
    /**
684  * Insert the method's description here.
     * Creation date: (14-10-00 13:35:55)
686  * @return javax.swing.JMenuItem
     */
688  public javax.swing.JMenuItem getIvjArrangeAllItem()
    {
690    if (ivjArrangeAllItem == null) {
       try {
692       ivjArrangeAllItem = new javax.swing.JMenuItem();
          ivjArrangeAllItem.setName("ArrangeAllItem");
694       ivjArrangeAllItem.setMnemonic('R');
          ivjArrangeAllItem.setText("Arrange␣all");
696       // user code begin {1}
          ivjArrangeAllItem.addActionListener( new PopUpActionListener() );
698       // user code end
       } catch (java.lang.Throwable ivjExc) {
700       // user code begin {2}
          // user code end
702       handleException(ivjExc);
       }
704    }
```

```
      return ivjArrangeAllItem;
706
   }
708 /**
    * Insert the method's description here.
710  * Creation date: (14-10-00 09:53:08)
    * @return javax.swing.JCheckBoxMenuItem
712  */
   public javax.swing.JCheckBoxMenuItem getIvjAutoLockAllItem() {
714   if (ivjAutoLockAllItem == null) {
         try {
716         ivjAutoLockAllItem = new javax.swing.JCheckBoxMenuItem();
            ivjAutoLockAllItem.setName("AutoLockAllItem");
718         ivjAutoLockAllItem.setMnemonic('A');
            ivjAutoLockAllItem.setText("Auto-lock␣all");
720         // user code begin {1}
            ivjAutoLockAllItem.addActionListener( new PopUpActionListener() );
722         // user code end
         } catch (java.lang.Throwable ivjExc) {
724         // user code begin {2}
            // user code end
726         handleException(ivjExc);
         }
728   }
      return ivjAutoLockAllItem;
730 }
   /**
732  * Insert the method's description here.
    * Creation date: (14-10-00 13:42:49)
734  * @return javax.swing.JCheckBoxMenuItem
    */
736 public javax.swing.JCheckBoxMenuItem getIvjLockItem()
   {
738   if (ivjLockItem == null) {
         try {
740         ivjLockItem = new javax.swing.JCheckBoxMenuItem();
            ivjLockItem.setName("LockItem");
742         ivjLockItem.setMnemonic('L');
            ivjLockItem.setText("Lock");
744         // user code begin {1}
            ivjLockItem.addActionListener( new PopUpActionListener() );
746         // user code end
         } catch (java.lang.Throwable ivjExc) {
748         // user code begin {2}
            // user code end
750         handleException(ivjExc);
         }
752   }
      return ivjLockItem;
754
   }
756 /**
    * Insert the method's description here.
758  * Creation date: (14-10-00 10:14:35)
    * @return javax.swing.JSeparator
760  */
   public javax.swing.JSeparator getIvjSeparator3() {
762   if (ivjSeparator3 == null) {
         try {
764         ivjSeparator3 = new javax.swing.JSeparator();
            ivjSeparator3.setName("JSeparator3");
766         // user code begin {1}
            // user code end
768      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
770         // user code end
```

```
            handleException(ivjExc);
772         }
         }
774      return ivjSeparator3;
      }
776   /**
       * Insert the method's description here.
778    * Creation date: (14-10-00 10:14:58)
       * @return javax.swing.JSeparator
780    */
      public javax.swing.JSeparator getIvjSeparator4() {
782      if (ivjSeparator4 == null) {
            try {
784            ivjSeparator4 = new javax.swing.JSeparator();
               ivjSeparator4.setName("JSeparator4");
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 ivjSeparator4;

796   }
      /**
798    * Insert the method's description here.
       * Creation date: (14-10-00 09:51:30)
800    * @return javax.swing.JMenuItem
       */
802   public javax.swing.JMenuItem getIvjUnlockAllItem() {
         if (ivjUnlockAllItem == null) {
804         try {
               ivjUnlockAllItem = new javax.swing.JMenuItem();
806            ivjUnlockAllItem.setName("UnlockAllItem");
               ivjUnlockAllItem.setMnemonic('N');
808            ivjUnlockAllItem.setText("Unlock␣all");
               // user code begin {1}
810            ivjUnlockAllItem.addActionListener( new PopUpActionListener() );
               // user code end
812         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
814            // user code end
               handleException(ivjExc);
816         }
         }
818      return ivjUnlockAllItem;

820   }
      /**
822    * Return the JSeparator2 property value.
       * @return javax.swing.JSeparator
824    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
826   private javax.swing.JSeparator getJSeparator2() {
         if (ivjJSeparator2 == null) {
828         try {
               ivjJSeparator2 = new javax.swing.JSeparator();
830            ivjJSeparator2.setName("JSeparator2");
               // user code begin {1}
832            // user code end
            } catch (java.lang.Throwable ivjExc) {
834            // user code begin {2}
               // user code end
836            handleException(ivjExc);
```

```
         }
      }
      return ivjJSeparator2;
   }
   /**
    * Insert the method's description here.
    * Creation date: (29-08-00 01:03:17)
    * @return clientapp.ClientPanoramaLayout
    */
   public ClientPanoramaLayoutManager getLayoutManager() {
      if( layoutManager == null )
      {
         setLayoutManager( new GridPanoramaLayoutManager() );
      }
      return layoutManager;
   }
   /**
    * Insert the method's description here.
    * Creation date: (26-07-00 11:38:26)
    * @return java.util.Hashtable
    */
   public java.util.Hashtable getLocalDocuments()
   {
      return localDocuments;
   }
   /**
    * Return the MoveItem property value.
    * @return javax.swing.JMenuItem
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getMoveItem() {
      if (ivjMoveItem == null) {
         try {
            ivjMoveItem = new javax.swing.JMenuItem();
            ivjMoveItem.setName("MoveItem");
            ivjMoveItem.setMnemonic('M');
            ivjMoveItem.setText("Move");
            // user code begin {1}
            ivjMoveItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjMoveItem;
   }
   /**
    * Insert the method's description here.
    * Creation date: (18-06-00 17:41:38)
    * @return clientapp.ClientApplication
    */
   public ClientApplication getOwner() {
      return owner;
   }
   /**
    * Return the ResizeItem property value.
    * @return javax.swing.JMenuItem
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getResizeItem() {
      if (ivjResizeItem == null) {
         try {
            ivjResizeItem = new javax.swing.JMenuItem();
            ivjResizeItem.setName("ResizeItem");
```

```
            ivjResizeItem.setMnemonic('R');
            ivjResizeItem.setText("Resize");
            // user code begin {1}
            ivjResizeItem.addActionListener( new PopUpActionListener() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjResizeItem;
   }
   /**
    * Insert the method's description here.
    * Creation date: (22-08-00 15:44:15)
    * @return javax.swing.Timer
    */
   public javax.swing.Timer getUpdateTimer()
   {
      if ( updateTimer == null)
      {  // 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,
                         new UpdateActionListener() );
      }
      return updateTimer;
   }
   /**
    * 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: (16-06-00 17:44:24)
    */
   public void initialize()
   {
      try {
         // user code begin {1}
         // user code end
         setName("ClientPanorama");
         setSize(1, 1);
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      // user code begin {2}
      getCameraNode().addMouseListener( new CameraMouseListener() );
      getCameraNode().addMouseMotionListener( new PanoramaMouseMotionListener() );
      getCamera().addCameraListener( new CameraListener() );
      zoomEventHandler = new PZoomHandler( getCameraNode() );
      zoomEventHandler.setActive( true );
      panEventHandler = new PZPanEventHandler( getCameraNode() );
      panEventHandler.setActive( true );
      swingEventHandler = new ZSwingEventHandler( this, getCameraNode() );
      swingEventHandler.setActive( true );
      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
```

```
1030   * @param args java.lang.String[]
       */
1032   public static void main(java.lang.String[] args) {
          try {
1034         JFrame frame = new javax.swing.JFrame();
             ClientPanorama aClientPanorama;
1036         aClientPanorama = new ClientPanorama();
             frame.setContentPane(aClientPanorama);
1038         frame.setSize(aClientPanorama.getSize());
             frame.addWindowListener(new java.awt.event.WindowAdapter() {
1040            public void windowClosing(java.awt.event.WindowEvent e) {
                  System.exit(0);
1042            };
             });
1044         frame.setVisible(true);
          } catch (Throwable exception) {
1046         System.err.println("Exception␣occurred␣in␣main()␣of␣clientapp.ClientPanorama");
             exception.printStackTrace(System.out);
1048      }
       }
1050   /**
       * Translate the document associated with the currently active node in the direction and by the amount
           indicated by the
1052   * user's mouse movements.
       * Creation date: (28-07-00 01:13:44)
1054   */
       public void moveDocument()
1056   {
          setMovingNode( true );
1058      previousMousePosition = null;
       }
1060   /**
       * Insert the method's description here.
1062   * Creation date: (22-08-00 11:18:54)
       * @return javax.swing.JTextArea
1064   * @param text java.lang.String
       */
1066   public JTextArea newDocumentLabel(String text)
       {
1068      JTextArea newLabel = new JTextArea( text );
          newLabel.setEditable( false );
1070      newLabel.setBackground( Color.orange );
       // newLabel.setSize( new Dimension( Integer.parseInt( ClientConstants.getDEFAULT_DOCUMENT_WIDTH() ), (
           int) newLabel.getSize().getHeight() ));
1072      Border bevelBorder = BorderFactory.createRaisedBevelBorder();
          Border matteBorder = BorderFactory.createMatteBorder( ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1074                                        ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                           ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1076                                       ClientConstants.getDOCUMENT_BORDER_WIDTH(),
                                           newLabel.getBackground() );
1078      Border compoundBorder = BorderFactory.createCompoundBorder( bevelBorder, matteBorder );
          newLabel.setBorder( compoundBorder );
1080      newLabel.setLineWrap( true );
          return newLabel;
1082   }
       /**
1084   * Insert the method's description here.
       * Creation date: (28-07-00 01:02:00)
1086   */
       public void popDownPopUpMenu()
1088   {
          getDocumentMenu().setVisible( false );
1090   // setCurrentlyClickedNode( null );
       }
1092   /**
```

```
     * Opens the document pop-up menu, modifying it to reflect the type of document on which the menu was
         invoked
1094 * Creation date: (20-07-00 16:48:11)
     */
1096 public void popUpPopUpMenu( ZVisualLeaf node, Point position )
     {
1098    setCurrentlyClickedNode( node );
       JScrollPane clickedPane = (JScrollPane) ( (PZSwing) node.getVisualComponent() ).getComponent();
1100   // enable text copying only if the clicked node displays text, either formatted or plain
       if ( clickedPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1102   {
          getCopyTextItem().setEnabled( true );
1104   }
       else
1106   {
          getCopyTextItem().setEnabled( false );
1108   }
       boolean state = ClientInfo.getPreferences().getProperty( ClientConstants.getAUTOLOCK_NODES() ).equals
            ( ClientConstants.getTRUE() )? true : false;
1110   getIvjAutoLockAllItem().setState( state );
       if (getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() )!= null )
1112   {
          state = getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() ).equals(
              ClientConstants.getTRUE() )? true : false;
1114   }
       else
1116   {
          state = false;
1118   }
       getIvjLockItem().setState( state );
1120   getDocumentMenu().show( this, (int)position.getX(), (int)position.getY() );
       getDocumentMenu().setVisible( true );
1122 }
     /**
1124  * Remove document originating from other group member.
      * Creation date: (03-08-00 13:13:00)
1126  * @param path java.lang.String
      */
1128 public void removeGlobalDocument(String senderName, String documentName )
     {
1130   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
       ZNode topNode = ( (ZNode) getDocuments().get( fqPath )).editor().getTop();
1132   if ( topNode != null )
       {
1134      topNode.getParent().removeChild( topNode );
       }
1136   owner.removeDocumentFromVisibleList( senderName, documentName );
       getDocuments().remove( fqPath );
1138 }
     /**
1140  * Remove document inserted by the user the client making the call to this method.
      * Creation date: (23-06-00 08:39:46)
1142  * @param path java.lang.String
      */
1144 public void removeLocalDocument(String senderName, String documentName )
     {
1146   String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
       ZNode topNode = ( (ZNode) getLocalDocuments().get( fqPath )).editor().getTop();
1148   if ( topNode != null )
       {
1150      topNode.getParent().removeChild( topNode );
          getLocalDocuments().remove( fqPath );
1152      // System.out.println( "Removed: " + fqPath );
       }
1154   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()
```

```
1586  {
         if ( ClientInfo.getPreferences().getProperty( ClientConstants.getZOOM_TO_OVERVIEW_ON_UPDATE() ).
1588         equalsIgnoreCase( ClientConstants.getZOOM_TO_OVERVIEW() ))
         {
1590        if ( (getDocuments().size() + getLocalDocuments().size()) > 0 )
            {
1592           java.awt.Rectangle targetRectangle = computeOverviewRectangle( getLayer().getChildren() );
               PZoomHandler.setTargetRectangle( targetRectangle );
1594           getCamera().center( targetRectangle,
                              Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
                                   getANIMATION_DURATION_IN_MILLISECS() )),
1596                          getDrawingSurface() );
               owner.getClientManager().clearTopicTree();
1598        }
         }
1600
      }
1602  }
```

Listing C.8: ClientPanoramaLayoutManager.java

```
   package clientapp;
 2
   import edu.umd.cs.jazz.*;
 4 import edu.umd.cs.jazz.event.*;
   import java.util.*;
 6 /**
    * The abstract root of the layout manager class hierarchy.
 8  * Creation date: (12-10-00 12:36:38)
    * @author:
10  */
   public abstract class ClientPanoramaLayoutManager implements ZLayoutManager {
12    private final static java.lang.String DOCUMENT_WIDTH = "Document␣width";
      private final static java.lang.String DOCUMENT_HEIGHT = "Document␣height";
14    private final static int DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
      private final static int DOCUMENT_WIDTH_DEFAULT_VALUE = 500;
16 /**
    * ClientPanoramaLayoutManager constructor comment.
18  */
   public ClientPanoramaLayoutManager() {
20    super();
   }
22 /**
    * Insert the method's description here.
24  * Creation date: (12-10-00 13:06:25)
    * @return java.lang.Object
26  */
   public Object clone() {
28    return null;
   }
30 /**
    * Insert the method's description here.
32  * Creation date: (12-10-00 23:47:58)
    * @param nodes edu.umd.cs.jazz.ZVisualLeaf[]
34  */
   public void doLayout(ZNode[] nodes)
36 {
   }
38 /**
    * doLayout method comment.
40  */
   public void doLayout(ZGroup node) {}
42 /**
    * Insert the method's description here.
44  * Creation date: (12-10-00 14:02:19)
    * @return java.lang.String
46  */
```

```
   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
{}
```

```
176 }
```

Listing C.9: ComposeNewMessage.java

```
   package clientapp;
 2
 4  import javax.swing.event.*;
    import java.awt.event.*;
 6  import peerviewmisc.*;
    import javax.swing.JMenuItem;
 8  import java.awt.datatransfer.*;

10  /**
    * This class implements the dialog box for composing new mesages
12  * Creation date: (10-07-00 23:54:05)
    * @author:
14  */
   public class ComposeNewMessage extends javax.swing.JDialog
16 {
      public class PopUpActionListener implements ActionListener
18    {
         public void actionPerformed( ActionEvent ae )
20       {
            if ( ae.getSource() instanceof JMenuItem )
22          {
               JMenuItem menuItem = (JMenuItem) ae.getSource();
24
               if ( menuItem == getPasteTextItem() )
26             {
                  pasteText();
28             }
            }
30       }
      };
32
      public class MouseHandler extends MouseAdapter
34    {
         public void mouseClicked( MouseEvent me )
36       {
            // if right-click
38          if ( me.isMetaDown() )
            {
40             popUpPopUpMenu( me.getPoint() );
            }
42       }
         public void mouseEntered( MouseEvent me )
44       {
            popDownPopUpMenu();
46       }
      };
48
      public class PopUpMouseHandler extends MouseAdapter
50    {
         public void mouseExited( MouseEvent me )
52       {
            popDownPopUpMenu();
54       }

56       public void mouseEntered( MouseEvent me )
         {
58          popDownPopUpMenu();
         }
60    };

62    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)
```

```
720   * @param newClientApplication clientapp.ClientApplication
      */
722   public void setClientApplication(ClientApplication newClientApplication) {
         clientApplication = newClientApplication;
724   }
      /**
726    * Insert the method's description here.
       * Creation date: (16-07-00 10:56:47)
728    * @param newDate java.util.GregorianCalendar
       */
730   public void setDate(java.util.Date newDate) {
         date = newDate;
732   }
      /**
734    * Insert the method's description here.
       * Creation date: (18-07-00 08:37:59)
736    * @param newMessage peerviewmisc.Message
       */
738   public void setMessage(peerviewmisc.Message newMessage) {
         message = newMessage;
740   }
      /**
742    * 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.
744    * Creation date: (11-07-00 01:11:45)
       */
746   public void submitAndClose()
      {
748      if ( getTitleField().getText().length() == 0 )
         {
750         MessageBox messageBox = new MessageBox();
            messageBox.setText( ClientConstants.getTITLE_REQUIRED_MESSAGE() );
752         messageBox.show();
            return;
754      }
         message = new peerviewmisc.Message();
756
         message.setAuthor( getAuthor() );
758      message.setContents( getContents() + ClientConstants.getCONTENTS_AND_SIGNATURE_DIVIDER() + ClientInfo
             .getPreferences().getProperty( ClientConstants.getSIGNATURE() ));
         message.setCreationDate( getDate() );
760      message.setTitle( getTitle() );
         dispose();
762   }
      }
```

Listing C.10: DiscussionFrame.java

```
      package clientapp;
 2
      import java.awt.*;
 4    import javax.swing.*;
      import javax.swing.border.*;
 6    import javax.swing.event.*;
      import javax.swing.tree.*;
 8    /**
       * This class implements the discussion panel in the bottom portion of the client application window.
10     * Creation date: (24-05-00 13:06:10)
       * @author:
12     */
      public class DiscussionFrame extends JFrame {
14       private JMenuItem ivjAboutItem = null;
         private JMenuItem ivjCloseWindowItem = null;
16       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();
```

```
84  }
    /**
86   * DiscussionFrame constructor comment.
     * @param title java.lang.String
88   */
    public DiscussionFrame(String title) {
90     super(title);
    }
92  /**
     * connEtoM1: (NewMessageItem.action.actionPerformed(java.awt.event.ActionEvent) --> ComposeContribution
         .show()V)
94   * @param arg1 java.awt.event.ActionEvent
     */
96  /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM1(java.awt.event.ActionEvent arg1) {
98     try {
          // user code begin {1}
100       // user code end
          getComposeContribution().show();
102       // user code begin {2}
          // user code end
104    } catch (java.lang.Throwable ivjExc) {
          // user code begin {3}
106       // user code end
          handleException(ivjExc);
108    }
    }
110 /**
     * connEtoM2: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) -->
         ComposeContribution.show()V)
112  * @param arg1 java.awt.event.ActionEvent
     */
114 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private void connEtoM2(java.awt.event.ActionEvent arg1) {
116    try {
          // user code begin {1}
118       // user code end
          getComposeContribution().show();
120       // user code begin {2}
          // user code end
122    } catch (java.lang.Throwable ivjExc) {
          // user code begin {3}
124       // user code end
          handleException(ivjExc);
126    }
    }
128 /**
     * Return the AboutItem property value.
130  * @return javax.swing.JMenuItem
     */
132 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenuItem getAboutItem() {
134    if (ivjAboutItem == null) {
          try {
136          ivjAboutItem = new javax.swing.JMenuItem();
             ivjAboutItem.setName("AboutItem");
138          ivjAboutItem.setText("About");
             // user code begin {1}
140          // user code end
          } catch (java.lang.Throwable ivjExc) {
142          // user code begin {2}
             // user code end
144          handleException(ivjExc);
          }
146    }
       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");
            ivjDiscussionMenu.add(getNewThreadItem());
            ivjDiscussionMenu.add(getNewMessageItem());
            ivjDiscussionMenu.add(getJSeparator1());
            ivjDiscussionMenu.add(getCloseWindowItem());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjDiscussionMenu;
   }
   /**
    * Return the DiscussionTree property value.
    * @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.setPreferredSize(new java.awt.Dimension(78, 200));
            ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjDiscussionTree;
   }
   /**
    * 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(getContentsItem());
            ivjHelpMenu.add(getAboutItem());
            // 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 JDialogContentPane property value.
    * @return javax.swing.JPanel
    */
   /* 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;
```

```
    }
870 /**
     * Return the JToolBarButton2 property value.
872  * @return javax.swing.JButton
     */
874 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JButton getJToolBarButton2() {
876    if (ivjJToolBarButton2 == null) {
          try {
878          ivjJToolBarButton2 = new javax.swing.JButton();
             ivjJToolBarButton2.setName("JToolBarButton2");
880          ivjJToolBarButton2.setText("");
             ivjJToolBarButton2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
882          ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
             ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Help16.gif")));
884          ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
             // user code begin {1}
886          // user code end
          } catch (java.lang.Throwable ivjExc) {
888          // user code begin {2}
             // user code end
890          handleException(ivjExc);
          }
892    }
       return ivjJToolBarButton2;
894 }
    /**
896  * Return the JToolBarButton3 property value.
     * @return javax.swing.JButton
898  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
900 private javax.swing.JButton getJToolBarButton3() {
       if (ivjJToolBarButton3 == null) {
902       try {
             ivjJToolBarButton3 = new javax.swing.JButton();
904          ivjJToolBarButton3.setName("JToolBarButton3");
             ivjJToolBarButton3.setText("");
906          ivjJToolBarButton3.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
             ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
908          ivjJToolBarButton3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                  toolbarButtonGraphics/general/Stop16.gif")));
             ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
910          // user code begin {1}
             // user code end
912       } catch (java.lang.Throwable ivjExc) {
             // user code begin {2}
914          // user code end
             handleException(ivjExc);
916       }
       }
918    return ivjJToolBarButton3;
    }
920 /**
     * Return the MessageDisplayArea property value.
922  * @return javax.swing.JTextPane
     */
924 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JTextPane getMessageDisplayArea() {
926    if (ivjMessageDisplayArea == null) {
          try {
928          ivjMessageDisplayArea = new javax.swing.JTextPane();
             ivjMessageDisplayArea.setName("MessageDisplayArea");
930          ivjMessageDisplayArea.setToolTipText("Message␣display␣area");
             ivjMessageDisplayArea.setEnabled(false);
932          // user code begin {1}
```

```
            ivjMessageDisplayArea.setBorder(BorderFactory.createEtchedBorder());
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjMessageDisplayArea;
   }
   /**
    * Return the MessageDisplayArea1 property value.
    * @return javax.swing.JTextPane
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextPane getMessageDisplayAreaOnComposeBox() {
      if (ivjMessageDisplayAreaOnComposeBox == null) {
         try {
            ivjMessageDisplayAreaOnComposeBox = new javax.swing.JTextPane();
            ivjMessageDisplayAreaOnComposeBox.setName("MessageDisplayAreaOnComposeBox");
            ivjMessageDisplayAreaOnComposeBox.setToolTipText("Message␣composition␣area");
            ivjMessageDisplayAreaOnComposeBox.setEnabled(true);
            // user code begin {1}
            ivjMessageDisplayAreaOnComposeBox.setBorder(BorderFactory.createEtchedBorder());
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjMessageDisplayAreaOnComposeBox;
   }
   /**
    * Return the MessagePanel property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getMessagePanel() {
      if (ivjMessagePanel == null) {
         try {
            ivjMessagePanel = new javax.swing.JPanel();
            ivjMessagePanel.setName("MessagePanel");
            ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
            ivjMessagePanel.add(getPanelHoldingHeaderOnComposeBox());
            getMessagePanel().add(getMessageDisplayAreaOnComposeBox(), getMessageDisplayAreaOnComposeBox().
                 getName());
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjMessagePanel;
   }
   /**
    * Return the MessagePanelBoxLayout property value.
    * @return javax.swing.BoxLayout
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.BoxLayout getMessagePanelBoxLayout() {
      javax.swing.BoxLayout ivjMessagePanelBoxLayout = null;
      try {
         /* Create part */
```

```
998         ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swing.BoxLayout.
                Y_AXIS);
      } catch (java.lang.Throwable ivjExc) {
1000     handleException(ivjExc);
      };
1002  return ivjMessagePanelBoxLayout;
   }
1004  /**
    * Return the NewMessageItem property value.
1006  * @return javax.swing.JMenuItem
    */
1008  /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getNewMessageItem() {
1010     if (ivjNewMessageItem == null) {
         try {
1012        ivjNewMessageItem = new javax.swing.JMenuItem();
           ivjNewMessageItem.setName("NewMessageItem");
1014       ivjNewMessageItem.setText("New␣message...");
           // user code begin {1}
1016       // user code end
         } catch (java.lang.Throwable ivjExc) {
1018       // user code begin {2}
           // user code end
1020       handleException(ivjExc);
         }
1022  }
      return ivjNewMessageItem;
1024  }
   /**
1026  * Return the NewThreadItem property value.
    * @return javax.swing.JMenuItem
1028  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
1030  private javax.swing.JMenuItem getNewThreadItem() {
      if (ivjNewThreadItem == null) {
1032     try {
           ivjNewThreadItem = new javax.swing.JMenuItem();
1034       ivjNewThreadItem.setName("NewThreadItem");
           ivjNewThreadItem.setText("New␣thread");
1036       // user code begin {1}
           // user code end
1038    } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
1040       // user code end
           handleException(ivjExc);
1042    }
      }
1044  return ivjNewThreadItem;
   }
1046  /**
    * Return the PanelHoldingHeader property value.
1048  * @return javax.swing.JPanel
    */
1050  /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getPanelHoldingHeader() {
1052     if (ivjPanelHoldingHeader == null) {
         try {
1054        ivjPanelHoldingHeader = new javax.swing.JPanel();
           ivjPanelHoldingHeader.setName("PanelHoldingHeader");
1056       ivjPanelHoldingHeader.setLayout(new javax.swing.BoxLayout(getPanelHoldingHeader(), javax.swing.
                BoxLayout.X_AXIS));
           ivjPanelHoldingHeader.setMaximumSize(new java.awt.Dimension(496, 120));
1058       ivjPanelHoldingHeader.setPreferredSize(new java.awt.Dimension(496, 70));
           ivjPanelHoldingHeader.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
1060       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
```

```
1308        } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
1310           // user code end
               handleException(ivjExc);
1312        }
         }
1314     return ivjSubjectLabel1;
      }
1316  /**
       * Return the JButton1 property value.
1318   * @return javax.swing.JButton
       */
1320  /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JButton getSubmitButton() {
1322     if (ivjSubmitButton == null) {
            try {
1324           ivjSubmitButton = new javax.swing.JButton();
               ivjSubmitButton.setName("SubmitButton");
1326           ivjSubmitButton.setText("Submit");
               // user code begin {1}
1328           // user code end
            } catch (java.lang.Throwable ivjExc) {
1330           // user code begin {2}
               // user code end
1332           handleException(ivjExc);
            }
1334     }
        return ivjSubmitButton;
1336  }
      /**
1338   * Called whenever the part throws an exception.
       * @param exception java.lang.Throwable
1340   */
      private void handleException(java.lang.Throwable exception) {
1342
        /* Uncomment the following lines to print uncaught exceptions to stdout */
1344    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
1346  }
      /**
1348   * Initializes connections
       * @exception java.lang.Exception The exception description.
1350   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1352  private void initConnections() throws java.lang.Exception {
        // user code begin {1}
1354    // user code end
        getNewMessageItem().addActionListener(ivjEventHandler);
1356    getJToolBarButton1().addActionListener(ivjEventHandler);
      }
1358  /**
       * Initialize the class.
1360   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
1362  private void initialize() {
        try {
1364       // user code begin {1}
           // user code end
1366       setName("DiscussionFrame");
           setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1368       setTitle("Discussion␣of␣xxx.yyy");
           setSize(600, 600);
1370       setJMenuBar(getDiscussionFrameJMenuBar());
           setContentPane(getJFrameContentPane());
1372       initConnections();
        } catch (java.lang.Throwable ivjExc) {
```

```
1374        handleException(ivjExc);
          }
1376      // user code begin {2}
          // user code end
1378    }
        /**
1380     * main entrypoint - starts the part when it is run as an application
         * @param args java.lang.String[]
1382     */
        public static void main(java.lang.String[] args) {
1384      try {
            DiscussionFrame aDiscussionFrame;
1386        aDiscussionFrame = new DiscussionFrame();
            aDiscussionFrame.addWindowListener(new java.awt.event.WindowAdapter() {
1388          public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
1390          };
            });
1392        aDiscussionFrame.setVisible(true);
          } catch (Throwable exception) {
1394        System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JFrame");
            exception.printStackTrace(System.out);
1396      }
        }
1398    }
```

Listing C.11: GridLayout.java

```
    package clientapp;
  2

  4  import java.awt.Dimension;
     import java.lang.Exception;
  6
     /**
  8  * 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)
 10  * @author:
     */
 12  public class GridLayout extends PanoramaLayout {
        private int width;
 14     private int height;
     /**
 16  * GridLayout constructor comment.
     */
 18  public GridLayout()
     {
 20     super();
        setHeight( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
            getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() )));
 22     setWidth( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
            getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() )));
     }
 24  /**
     * Convert the vertical and horizontal indices into a x,y coordinate pair designating a position in the
         panorama.
 26  * Creation date: (22-06-00 12:02:00)
     * @return java.awt.Dimension
 28  */
     public java.awt.Dimension computePlacement() throws Exception
 30  {
        Dimension newPlacement = new Dimension( (int) ( horizontalIndex * (int)(preferredDocumentSize.
            getWidth() + getXSpacing() )),
 32                                 (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.
```

```
100   * Creation date: (22-06-00 11:43:44)
      * @param newHeight int
102   */
      public void setHeight(int newHeight) {
104     height = newHeight;
      }
106   /**
      * Insert the method's description here.
108   * Creation date: (22-06-00 11:43:29)
      * @param newWidth int
110   */
      public void setWidth(int newWidth) {
112     width = newWidth;
      }
114   }
```

Listing C.12: GridPanoramaLayoutManager.java

```
      package clientapp;
 2
       import java.util.ArrayList;
 4     import java.awt.event.*;
       import javax.swing.event.*;
 6
       /**
 8     * This class implements a simple layout manager which arranges the panorama documents in matrix pattern
          .
       * Creation date: (28-08-00 23:56:47)
10     * @author:
       */
12    public class GridPanoramaLayoutManager extends ClientPanoramaLayoutManager
      {
14      private javax.swing.JPanel ivjButtonPanel = null;
        private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
16      private javax.swing.JButton ivjCancelButton = null;
        private javax.swing.JTextField ivjDocumentHeightField = null;
18      private javax.swing.JLabel ivjDocumentHeightLabel = null;
        private javax.swing.JTextField ivjDocumentWidthField = null;
20      private javax.swing.JLabel ivjDocumentWidthLabel = null;
        private javax.swing.JDialog ivjGridLayoutCustomizationBox = null;
22      private javax.swing.JTextField ivjHorizontalSpacingField = null;
        private javax.swing.JLabel ivjHorizontalSpacingLabel = null;
24      private javax.swing.JPanel ivjJDialogContentPane = null;
        private javax.swing.JPanel ivjMainPanel = null;
26      private javax.swing.JButton ivjOKButton = null;
        private javax.swing.JTextField ivjVerticalSpacingField = null;
28      private javax.swing.JLabel ivjVerticalSpacingLabel = null;
        private final static java.lang.String GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING = "Grid␣layout␣document
            ␣horizontal␣spacing";
30      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;
32      private final static int GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE = 200;

34      public class CancelHandler implements ActionListener
        {
36        public void actionPerformed( ActionEvent ae )
          {
38          setCommitted( false );
            getGridLayoutCustomizationBox().dispose();
40        }
        };
42
        public class OKHandler implements ActionListener
44      {
          public void actionPerformed( ActionEvent ae )
46        {
```

```
            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() {
```

```
372     if (ivjGridLayoutCustomizationBox == null) {
          try {
374         ivjGridLayoutCustomizationBox = new javax.swing.JDialog();
            ivjGridLayoutCustomizationBox.setName("GridLayoutCustomizationBox");
376         ivjGridLayoutCustomizationBox.setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
            ivjGridLayoutCustomizationBox.setBounds(24, 44, 398, 300);
378         ivjGridLayoutCustomizationBox.setModal(true);
            ivjGridLayoutCustomizationBox.setTitle("Grid␣layout␣customization");
380         getGridLayoutCustomizationBox().setContentPane(getJDialogContentPane());
            // user code begin {1}
382         // user code end
          } catch (java.lang.Throwable ivjExc) {
384         // user code begin {2}
            // user code end
386         handleException(ivjExc);
          }
388     }
        return ivjGridLayoutCustomizationBox;
390   }
      /**
392    * Return the HorizontalSpacingField property value.
       * @return javax.swing.JTextField
394    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
396   private javax.swing.JTextField getHorizontalSpacingField() {
        if (ivjHorizontalSpacingField == null) {
398       try {
            ivjHorizontalSpacingField = new javax.swing.JTextField();
400         ivjHorizontalSpacingField.setName("HorizontalSpacingField");
            // user code begin {1}
402         // user code end
          } catch (java.lang.Throwable ivjExc) {
404         // user code begin {2}
            // user code end
406         handleException(ivjExc);
          }
408     }
        return ivjHorizontalSpacingField;
410   }
      /**
412    * Return the HorizontalSpacingLabel property value.
       * @return javax.swing.JLabel
414    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
416   private javax.swing.JLabel getHorizontalSpacingLabel() {
        if (ivjHorizontalSpacingLabel == null) {
418       try {
            ivjHorizontalSpacingLabel = new javax.swing.JLabel();
420         ivjHorizontalSpacingLabel.setName("HorizontalSpacingLabel");
            ivjHorizontalSpacingLabel.setText("Horizontal␣spacing:");
422         // user code begin {1}
            // user code end
424       } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
426         // user code end
            handleException(ivjExc);
428       }
        }
430     return ivjHorizontalSpacingLabel;
      }
432   /**
       * Return the JDialogContentPane property value.
434    * @return javax.swing.JPanel
       */
436   /* 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
```

```
     */
568  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JTextField getVerticalSpacingField() {
570     if (ivjVerticalSpacingField == null) {
           try {
572           ivjVerticalSpacingField = new javax.swing.JTextField();
              ivjVerticalSpacingField.setName("VerticalSpacingField");
574           // user code begin {1}
              // user code end
576        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
578           // user code end
              handleException(ivjExc);
580        }
        }
582     return ivjVerticalSpacingField;
     }
584  /**
      * Return the VerticalSpacingLabel property value.
586   * @return javax.swing.JLabel
      */
588  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JLabel getVerticalSpacingLabel() {
590     if (ivjVerticalSpacingLabel == null) {
           try {
592           ivjVerticalSpacingLabel = new javax.swing.JLabel();
              ivjVerticalSpacingLabel.setName("VerticalSpacingLabel");
594           ivjVerticalSpacingLabel.setText("Vertical␣spacing:");
              // 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 ivjVerticalSpacingLabel;
604  }
     /**
606   * Called whenever the part throws an exception.
      * @param exception java.lang.Throwable
608   */
     private void handleException(java.lang.Throwable exception) {
610
        /* Uncomment the following lines to print uncaught exceptions to stdout */
612     // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
614  }
     /**
616   * Initialize the class.
      */
618  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
620     try {
           // user code begin {1}
622        // user code end
        } catch (java.lang.Throwable ivjExc) {
624        handleException(ivjExc);
        }
626     // user code begin {2}
        // user code end
628  }
     /**
630   * Insert the method's description here.
      * Creation date: (12-10-00 15:31:05)
632   * @return boolean
```

```
     */
634  public boolean isCommitted() {
       return committed;
636  }
     /**
638   * main entrypoint - starts the part when it is run as an application
      * @param args java.lang.String[]
640   */
     public static void main(java.lang.String[] args) {
642     try {
          GridPanoramaLayoutManager aGridPanoramaLayoutManager;
644       aGridPanoramaLayoutManager = new GridPanoramaLayoutManager();
       } catch (Throwable exception) {
646       System.err.println("Exception␣occurred␣in␣main()␣of␣clientapp.ClientPanoramaLayout");
          exception.printStackTrace(System.out);
648    }
     }
650  /**
      * Insert the method's description here.
652   * Creation date: (12-10-00 15:31:05)
      * @param newCommitted boolean
654   */
     public void setCommitted(boolean newCommitted) {
656    committed = newCommitted;
     }
658  /**
      * Set document specific preferences.
660   * Creation date: (12-10-00 13:46:17)
      * @param newSettings java.util.Properties
662   */
     public void setSettings(java.util.Properties newSettings)
664  {
       if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() )!= null )
666    {
          getDocumentHeightField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() ));
668    }
       else
670    {
          getDocumentHeightField().setText( String.valueOf( getDOCUMENT_HEIGHT_DEFAULT_VALUE() ));
672    }
       if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() )!= null )
674    {
          getDocumentWidthField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() ));
676    }
       else
678    {
          getDocumentWidthField().setText( String.valueOf( getDOCUMENT_WIDTH_DEFAULT_VALUE() ));
680    }
       if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() )!= null )
682    {
          getHorizontalSpacingField().setText( newSettings.getProperty(
               getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() ));
684    }
       else
686    {
          getHorizontalSpacingField().setText( String.valueOf(
               getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ));
688    }
       if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() )!= null )
690    {
          getVerticalSpacingField().setText( newSettings.getProperty(
               getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() ));
692    }
       else
694    {
```

```
      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();
         }
      }

      public class JoinGroupButtonHandler implements ActionListener
      {
         public void actionPerformed ( ActionEvent ae )
         {
            joinGroup();
         }
      }

      public class CloseButtonHandler implements ActionListener
      {
         public void actionPerformed( ActionEvent ae )
         {
            dispose();
         }
      }
      private javax.swing.JButton ivjCloseButton = null;
      private javax.swing.JButton ivjCreateGroupButton = null;
      private javax.swing.JButton ivjDeleteGroupButton = null;
      private javax.swing.JPanel ivjJDialogContentPane = null;
      private javax.swing.JButton ivjJoinGroupButton = null;
      private javax.swing.JPanel ivjJPanel2 = null;
      private java.awt.FlowLayout ivjJPanel2FlowLayout = null;
      private javax.swing.JPanel ivjJPanel3 = null;
      private java.awt.FlowLayout ivjJPanel3FlowLayout = null;
      private javax.swing.JScrollPane ivjJScrollPane1 = null;
      private javax.swing.JTable ivjScrollPaneTable = null;
      private DeleteGroup ivjDeleteGroup1 = null;
      private NewGroup ivjNewGroup1 = null;
      private peerviewmisc.WorkGroup workGroup = null;
      private EditGroup ivjEditGroup1 = null;
      private javax.swing.JButton ivjEditGroupButton = null;
      private javax.swing.table.DefaultTableModel tableModel;
      private javax.swing.JPanel ivjButtonPanel = null;
      private javax.swing.JPanel ivjTablePanel = null;
      private peerviewmisc.WorkGroup[] groupDirectoryValues = null;
      IvjEventHandler ivjEventHandler = new IvjEventHandler();
      private ClientApplication clientApplication = null;

   class IvjEventHandler implements java.awt.event.ActionListener {
         public void actionPerformed(java.awt.event.ActionEvent e) {
            if (e.getSource() == GroupDirectory.this.getJoinGroupButton())
               connEtoM1(e);
            if (e.getSource() == GroupDirectory.this.getCreateGroupButton())
               connEtoM2(e);
            if (e.getSource() == GroupDirectory.this.getEditGroupButton())
               connEtoM3(e);
            if (e.getSource() == GroupDirectory.this.getDeleteGroupButton())
               connEtoM4(e);
            if (e.getSource() == GroupDirectory.this.getCloseButton())
               connEtoM5(e);
         };
      };
   /**
    * GroupDirectory constructor comment.
    */
   public GroupDirectory() {
      super();
      initialize();
   }
   /**
    * GroupDirectory constructor comment.
    * @param owner java.awt.Dialog
```

```
 */
public GroupDirectory(java.awt.Dialog owner) {
   super(owner);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public GroupDirectory(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public GroupDirectory(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public GroupDirectory(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 */
public GroupDirectory(java.awt.Frame owner) {
   super(owner);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public GroupDirectory(java.awt.Frame owner, String title) {
   super(owner, title);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public GroupDirectory(java.awt.Frame owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * GroupDirectory constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public GroupDirectory(java.awt.Frame owner, boolean modal) {
   super(owner, modal);
}
/**
 * connEtoM1: (JoinGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
     joinGroup()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.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) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM5: (CloseButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM5(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);
   }
}
/**
 * Display a dialog box for entering information about a new group and create a corresponding object if the user acknowledges
 * his or her entries.
 * Creation date: (03-07-00 21:44:35)
 */
public void createNewGroup()
{
   NewGroup newGroup = getNewGroup1();

   newGroup.setWorkGroup( new WorkGroup() );
   newGroup.show();
   WorkGroup workGroup = newGroup.getWorkGroup();
   // if the workgroup box wasn't cancelled
   if ( newGroup.isIsDiscarded() == false )
   {
      // complete the non-editable fields of the new workgroup
      workGroup.setCreationDate( new java.util.Date() );
      workGroup.setCreator( (String) ClientInfo.getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ));
      getClientApplication().getClientManager().createWorkGroup( workGroup );
   }
}
/**
 * Delete whatever group is currently selected in the group directory listing if the user acknowledges the deletion.
 * Creation date: (03-07-00 21:45:09)
 */
public void deleteGroup()
{
   int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
   // if no selection
   if ( selectedRow == -1 )
   {
      MessageBox messageBox = new MessageBox();
      messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_DELETE() );
      messageBox.show();
   }
   else
   {  WorkGroup selectedGroup = (WorkGroup) groupDirectoryValues[ selectedRow ];
      // if the selected group has active participants
```

```
      if ( selectedGroup.getParticipants().size() > 0 )
      {
         MessageBox messageBox = new MessageBox();
         messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS());
         messageBox.show();
         return;
      }
      ConfirmationBox confirmationBox = new ConfirmationBox();
      confirmationBox.setText( ClientConstants.getCONFIRM_WORKGROUP_DELETION( selectedGroup.getGroupName
          () ));
      confirmationBox.show();
      if ( confirmationBox.isConfirmed() == true )
      {
         getClientApplication().getClientManager().removeWorkGroup( selectedGroup );
      }
   }
}
/**
 * Disable all buttons that trigger actions involving communication with the server. Used in case of an
     inactive server
 * or a broken communications channel.
 * Creation date: (10-08-00 16:13:10)
 */
private void disableButtons()
{
   getJoinGroupButton().setEnabled( false);
   getCreateGroupButton().setEnabled( false );
   getEditGroupButton().setEnabled( false );
   getDeleteGroupButton().setEnabled( false );
}
/**
 * Display an edit box containing the attributes of the group currently selected in the group directory.
 * Creation date: (03-07-00 21:44:52)
 */
public void editExistingGroup()
{
   int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
   // if no selection
   if ( selectedRow == -1 )
   {
      MessageBox messageBox = new MessageBox();
      messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() );
      messageBox.show();
   }
   else
   {
      WorkGroup backup = new WorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );

      // if the selected group has active participants
      if ( backup.getParticipants().size() > 0 )
      {
         MessageBox messageBox = new MessageBox();
         messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() );
         messageBox.show();
         return;
      }
      getEditGroup1().setWorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
      getEditGroup1().show();
      if ( getEditGroup1().isIsDiscarded() == false )
      {
         getClientApplication().getClientManager().removeWorkGroup( backup );
         WorkGroup newWorkGroup = getEditGroup1().getWorkGroup();
         newWorkGroup.setCreationDate( new java.util.Date() );
         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");
```

```
         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 CreateGroupButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCreateGroupButton() {
   if (ivjCreateGroupButton == null) {
      try {
         ivjCreateGroupButton = new javax.swing.JButton();
         ivjCreateGroupButton.setName("CreateGroupButton");
         ivjCreateGroupButton.setMnemonic('g');
         ivjCreateGroupButton.setText("Create␣group");
         // user code begin {1}
         //ivjCreateGroupButton.addActionListener( new CreateGroupButtonHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjCreateGroupButton;
}
/**
 * Return the DeleteGroup1 property value.
 * @return peerviewmisc.DeleteGroup
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public peerviewmisc.DeleteGroup getDeleteGroup1() {
   if (ivjDeleteGroup1 == null) {
      try {
         ivjDeleteGroup1 = new peerviewmisc.DeleteGroup();
         ivjDeleteGroup1.setName("DeleteGroup1");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjDeleteGroup1;
}
/**
 * Return the DeleteGroupButton property value.
 * @return javax.swing.JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getDeleteGroupButton() {
   if (ivjDeleteGroupButton == null) {
      try {
         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;
    }
```

```
564  /**
      * Return the JDialogContentPane property value.
566   * @return javax.swing.JPanel
      */
568  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJDialogContentPane() {
570     if (ivjJDialogContentPane == null) {
           try {
572           ivjJDialogContentPane = new javax.swing.JPanel();
              ivjJDialogContentPane.setName("JDialogContentPane");
574           ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
              getJDialogContentPane().add(getTablePanel(), "Center");
576           getJDialogContentPane().add(getButtonPanel(), "South");
              // user code begin {1}
578           // user code end
           } catch (java.lang.Throwable ivjExc) {
580           // user code begin {2}
              // user code end
582           handleException(ivjExc);
           }
584     }
        return ivjJDialogContentPane;
586  }
     /**
588   * Return the JoinGroupButton property value.
      * @return javax.swing.JButton
590   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
592  private javax.swing.JButton getJoinGroupButton() {
        if (ivjJoinGroupButton == null) {
594        try {
              ivjJoinGroupButton = new javax.swing.JButton();
596           ivjJoinGroupButton.setName("JoinGroupButton");
              ivjJoinGroupButton.setMnemonic('J');
598           ivjJoinGroupButton.setText("Join␣group");
              // user code begin {1}
600           //ivjJoinGroupButton.addActionListener( new JoinGroupButtonHandler() );
              // user code end
602        } catch (java.lang.Throwable ivjExc) {
              // user code begin {2}
604           // user code end
              handleException(ivjExc);
606        }
        }
608     return ivjJoinGroupButton;
     }
610  /**
      * Return the JPanel2 property value.
612   * @return javax.swing.JPanel
      */
614  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private javax.swing.JPanel getJPanel2() {
616     if (ivjJPanel2 == null) {
           try {
618           ivjJPanel2 = new javax.swing.JPanel();
              ivjJPanel2.setName("JPanel2");
620           ivjJPanel2.setLayout(getJPanel2FlowLayout());
              getJPanel2().add(getJoinGroupButton(), getJoinGroupButton().getName());
622           getJPanel2().add(getCreateGroupButton(), getCreateGroupButton().getName());
              ivjJPanel2.add(getEditGroupButton());
624           ivjJPanel2.add(getDeleteGroupButton());
              // user code begin {1}
626           // user code end
           } catch (java.lang.Throwable ivjExc) {
628           // 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;
762   }
      /**
764    * Insert the method's description here.
       * Creation date: (04-07-00 19:38:32)
766    * @return javax.swing.table.DefaultTableModel
       */
768   public javax.swing.table.DefaultTableModel getTableModel() {
      return tableModel;
770   }
      /**
772    * Return the JPanel4 property value.
       * @return javax.swing.JPanel
774    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
776   private javax.swing.JPanel getTablePanel() {
      if (ivjTablePanel == null) {
778      try {
            ivjTablePanel = new javax.swing.JPanel();
780         ivjTablePanel.setName("TablePanel");
            ivjTablePanel.setLayout(new java.awt.BorderLayout());
782         getTablePanel().add(getJScrollPane1(), "Center");
            // user code begin {1}
784         getTablePanel().setBorder( ClientConstants.createDialogBorder( getTablePanel().getBackground() )
                 );
            // user code end
786      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
788         // user code end
            handleException(ivjExc);
790      }
      }
792   return ivjTablePanel;
      }
794   /**
       * Insert the method's description here.
796    * Creation date: (03-07-00 21:00:14)
       * @return peerviewmisc.WorkGroup
798    */
      public peerviewmisc.WorkGroup getWorkGroup() {
800   return workGroup;
      }
802   /**
       * Traverses the group directory (list of active groups) to determine if any group matches the group
           name and description parameters
804    * Creation date: (26-07-00 16:59:38)
       * @return boolean
806    * @param groupName java.lang.String
       * @param description java.lang.String
808    */
      public boolean groupExists(String groupName, String description)
810   {
      boolean exists = false;
812   for ( int i = 0; i < groupDirectoryValues.length; i++ )
      {
814      if ( ( (WorkGroup) groupDirectoryValues[i]).getGroupName().equalsIgnoreCase( groupName )&&
             ( (WorkGroup) groupDirectoryValues[i]).getDescription().equalsIgnoreCase( description ))
816      {
            exists = true;
818         break;
         }
820   }
      return exists;
822   }
      /**
824    * 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()
892 {
   }
894 /**
    * Insert the method's description here.
896  * Creation date: (06-07-00 12:48:00)
    */
898 public void joinGroup()
   {
900    int selectedIndex = getScrollPaneTable().getSelectedRow();
       // if no row is selected
902    if ( selectedIndex == -1 )
       {
904       MessageBox messageBox = new MessageBox();
          messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_JOIN_MESSAGE() );
906       messageBox.show();
          return;
908    }
       WorkGroup workGroup = (WorkGroup) groupDirectoryValues[ selectedIndex ];
910    URLString url = workGroup.getSessionURL();
       Session session;
912    Channel channel;
       String clientNames[] = null;
914    try
       {
916       session = SessionFactory.createSession( getClientApplication().getClientManager().
             getGroupManagementClient(), url, false );
          channel = session.createChannel( getClientApplication().getClientManager().
             getGroupManagementClient(),
918                                 workGroup.getWorkGroupChannelID(),
                                    true, true, false);
920       clientNames = channel.listClientNames();
       }
922    // 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.
924    catch ( NameInUseException NIUE )
       {
926       MessageBox messageBox = new MessageBox();
          messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
928       messageBox.show();
          return;
930    }
       catch ( JSDTException JSDTE )
932    {
          getClientApplication().getClientManager().handleJSDTException( JSDTE );
934    }
       for ( int i = 0 ; i < clientNames.length; i++ )
936    {
          if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
             getNAME() )))
938       {
             MessageBox messageBox = new MessageBox();
940          messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
             messageBox.show();
942          return;
          }
944    }
   /* for ( int i = 0; i < clientNames.length; i++ )
946    {
          // System.out.println( "Clientname: " + clientNames[i] );
948       // if client is already joined to the selected group
          if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
             getNAME() )));
950       {
          }
```

```
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 )
       {
```

```
1018       disableButtons();
           getClientApplication().getClientManager().requestGroupsFromServer();
1020       return;
        }
1022    enableButtons();
        // clear the table
1024    tableModel.setNumRows( 0 );
        // update with contents of group directory
1026    Collection values = getClientApplication().getClientManager().getClientInfo().getGroups().values();
        Iterator iterator = values.iterator();
1028    groupDirectoryValues = new WorkGroup[ values.size() ];
        int i = 0;
1030    while ( iterator.hasNext() )
        {
1032       workGroup = (WorkGroup) iterator.next();
           Vector v = workGroup.getGroupAsVector();
1034       tableModel.addRow( v );
           groupDirectoryValues[i++] = workGroup;
1036    }
        // Distribute surplus space evenly
1038    getScrollPaneTable().sizeColumnsToFit( -1 );
        getScrollPaneTable().revalidate();
1040    getScrollPaneTable().repaint();
        getJScrollPane1().revalidate();
1042    getJScrollPane1().repaint();

1044 }
     }
```

Listing C.14: MessagePropertiesBox.java

```
     package clientapp;
 2
      import javax.swing.table.DefaultTableModel;
 4    import java.util.Vector;
      import java.awt.event.ActionListener;
 6
      /**
 8    * 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.
10    * Creation date: (19-07-00 17:55:27)
      * @author:
12    */
     public class MessagePropertiesBox extends javax.swing.JDialog {
14
        public class CloseButtonHandler implements ActionListener
16      {
           public void actionPerformed( java.awt.event.ActionEvent ae )
18         {
              dispose();
20         }
        }
22      private javax.swing.JPanel ivjJDialogContentPane = null;
        private javax.swing.JScrollPane ivjJScrollPane1 = null;
24      private javax.swing.JTable ivjScrollPaneTable = null;
        private javax.swing.JButton ivjCloseButton = null;
26      private javax.swing.table.DefaultTableModel tableModel = new DefaultTableModel();
        private javax.swing.JPanel ivjButtonPanel = null;
28      private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
     /**
30    * MessagePropertiesBox constructor comment.
      */
32   public MessagePropertiesBox() {
        super();
34      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}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjScrollPaneTable;
   }
   /**
    * Insert the method's description here.
    * Creation date: (19-07-00 19:46:21)
    * @return javax.swing.table.DefaultTableModel
    */
   public javax.swing.table.DefaultTableModel getTableModel() {
      return tableModel;
   }
   /**
    * 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("MessagePropertiesBox");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         setSize(415, 257);
         setModal(true);
         setTitle("Message properties");
         setContentPane(getJDialogContentPane());
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      }
      // user code begin {2}
      // default positioning puts the dialog at center of screen. The method used is a slight hack, but
           works fine.
      setLocationRelativeTo( getOwner() );
      getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
      // user code end
   }
   /**
    * Insert the method's description here.
    * Creation date: (19-07-00 19:45:02)
    * @param message peerviewmisc.Message
    */
   public void initTable(peerviewmisc.Message message)
   {
      // 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();
      Vector values = message.getDataAsVector();
      tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_NAME_COLUMN_HEADING() );
      tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADING() );
      for ( int i = 0; i < fieldNames.length; i++ )
      {
         Vector v = new Vector();
         v.addElement( fieldNames[i] );
```

```
         v.addElement( values.get(i) );
298      tableModel.addRow( v );
      }
300   getScrollPaneTable().setColumnSelectionAllowed( false );
      getScrollPaneTable().setRowSelectionAllowed( false );
302   getScrollPaneTable().setEnabled( false );
      getScrollPaneTable().setCellEditor( null );
304   getScrollPaneTable().removeEditor();
   }
306 /**
    * main entrypoint - starts the part when it is run as an application
308 * @param args java.lang.String[]
    */
310 public static void main(java.lang.String[] args) {
      try {
312      MessagePropertiesBox aMessagePropertiesBox;
         aMessagePropertiesBox = new MessagePropertiesBox();
314      aMessagePropertiesBox.setModal(true);
         aMessagePropertiesBox.addWindowListener(new java.awt.event.WindowAdapter() {
316         public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
318         };
         });
320      aMessagePropertiesBox.setVisible(true);
      } catch (Throwable exception) {
322      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
324   }
   }
326 /**
    * Insert the method's description here.
328 * Creation date: (19-07-00 19:46:21)
    * @param newTableModel javax.swing.table.DefaultTableModel
330 */
   public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
332   tableModel = newTableModel;
   }
334 }
```

Listing C.15: MessageWindow.java

```
   package clientapp;
 2

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

 6  /**
    * This class implements the message window that can be activated by double clicking in the message bar
        at the bottom of the
 8  * client application window.
    * Creation date: (16-08-00 16:44:45)
10  * @author:
    */
12 public class MessageWindow extends javax.swing.JDialog {

14    public class CloseItemHandler implements ActionListener
      {
16       public void actionPerformed( ActionEvent ae )
         {
18          closeMessageWindow();
         }
20    }

22    public class SaveItemHandler implements ActionListener
      {
24       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);
    }
```

```
92  /**
     * MessageWindow constructor comment.
94   * @param owner java.awt.Frame
     * @param title java.lang.String
96   * @param modal boolean
     */
98  public MessageWindow(java.awt.Frame owner, String title, boolean modal) {
       super(owner, title, modal);
100 }
    /**
102  * MessageWindow constructor comment.
     * @param owner java.awt.Frame
104  * @param modal boolean
     */
106 public MessageWindow(java.awt.Frame owner, boolean modal) {
       super(owner, modal);
108 }
    /**
110  * Insert the method's description here.
     * Creation date: (16-08-00 17:13:28)
112  * @param string java.lang.String
     */
114 public void addString(String string)
    {
116    getTextArea().append( string + "\n" );
    }
118 /**
     * Insert the method's description here.
120  * Creation date: (16-08-00 17:51:13)
     */
122 public void closeMessageWindow()
    {
124    setVisible( false );
    }
126 /**
     * Insert the method's description here.
128  * Creation date: (29-10-00 17:08:48)
     * @return clientapp.ClientApplication
130  */
    public ClientApplication getClientApplication() {
132    return clientApplication;
    }
134 /**
     * Return the CloseItem property value.
136  * @return javax.swing.JMenuItem
     */
138 /* WARNING: THIS METHOD WILL BE REGENERATED. */
    private javax.swing.JMenuItem getCloseItem() {
140    if (ivjCloseItem == null) {
          try {
142          ivjCloseItem = new javax.swing.JMenuItem();
             ivjCloseItem.setName("CloseItem");
144          ivjCloseItem.setMnemonic('C');
             ivjCloseItem.setText("Close");
146          // user code begin {1}
             ivjCloseItem.addActionListener( new CloseItemHandler() );
148          // user code end
          } catch (java.lang.Throwable ivjExc) {
150          // user code begin {2}
             // user code end
152          handleException(ivjExc);
          }
154    }
       return ivjCloseItem;
156 }
    /**
```

```
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;
```

```
}
/**
 * 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 MessageWindowJMenuBar property value.
 * @return javax.swing.JMenuBar
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuBar getMessageWindowJMenuBar() {
   if (ivjMessageWindowJMenuBar == null) {
      try {
         ivjMessageWindowJMenuBar = new javax.swing.JMenuBar();
         ivjMessageWindowJMenuBar.setName("MessageWindowJMenuBar");
         ivjMessageWindowJMenuBar.add(getFileMenu());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjMessageWindowJMenuBar;
}
/**
 * Return the SaveItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getSaveItem() {
   if (ivjSaveItem == null) {
      try {
         ivjSaveItem = new javax.swing.JMenuItem();
         ivjSaveItem.setName("SaveItem");
         ivjSaveItem.setMnemonic('S');
         ivjSaveItem.setText("Save");
         // user code begin {1}
         ivjSaveItem.addActionListener( new SaveItemHandler() );
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjSaveItem;
}
/**
```

```
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
```

```
    {
20     public void stateChanged( ChangeEvent ce )
       {
22        // System.out.println( getAnimationSpeedSlider().getValue() );
       }
24  };

26  public class OKHandler implements ActionListener
    {
28     public void actionPerformed (ActionEvent ae)
       {
30        submitAndClose();
       }
32  }

34  public class CancelHandler implements ActionListener
    {
36     public void actionPerformed (ActionEvent ae)
       {
38        cancelAndClose();
       }
40  }

42  public class ConnectHandler implements ActionListener
    {
44     public void actionPerformed (ActionEvent ae)
       {
46        connectToServer();
       }
48  }

50  public class CustomizeHandler implements ActionListener
    {
52     public void actionPerformed( ActionEvent ae )
       {
54        customizeLayout();
       }
56  };

58  private JPanel ivjJDialogContentPane = null;
    private JTabbedPane ivjJTabbedPane1 = null;
60  private JPanel ivjDisplayPage = null;
    private JLabel ivjNameLabel = null;
62  private JPanel ivjPersonalPage = null;
    private JLabel ivjSignatureLabel = null;
64  private JButton ivjCancelButton = null;
    private JPanel ivjJPanel1 = null;
66  private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
    private JButton ivjOKButton = null;
68  private JScrollPane ivjJScrollPane1 = null;
    private ClientApplication clientApplication = null;
70  private JButton ivjCustomizeButton = null;
    private JComboBox ivjDisplayQualityBox = null;
72  private JLabel ivjDisplayQualityLabel = null;
    private JComboBox ivjLayoutSchemeBox = null;
74  private JLabel ivjLayoutSchemeLabel = null;
    IvjEventHandler ivjEventHandler = new IvjEventHandler();
76  private JTextField ivjNameTextField = null;
    private JTextPane ivjSignaturePane = null;
78  private JLabel ivjServerName = null;
    private JPanel ivjServerPage = null;
80  private JLabel ivjServerPortLabel = null;
    private JTextField ivjServerNameField = null;
82  private JTextField ivjServerPortField = null;
    private JLabel ivjConnectionType = null;
84  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 {
348      ivjConnectionTypeBox = new javax.swing.JComboBox();
         ivjConnectionTypeBox.setName("ConnectionTypeBox");
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 ivjConnectionTypeBox;
}
360 /**
 * Return the CustomizeButton property value.
362 * @return javax.swing.JButton
 */
364 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getCustomizeButton() {
366 if (ivjCustomizeButton == null) {
      try {
368      ivjCustomizeButton = new javax.swing.JButton();
         ivjCustomizeButton.setName("CustomizeButton");
370      ivjCustomizeButton.setMnemonic('u');
         ivjCustomizeButton.setText("Customize");
372      // user code begin {1}
         ivjCustomizeButton.addActionListener( new CustomizeHandler() );
374      // user code end
      } catch (java.lang.Throwable ivjExc) {
376      // user code begin {2}
         // user code end
378      handleException(ivjExc);
      }
380 }
   return ivjCustomizeButton;
382 }
   /**
384 * Return the DisplayPage property value.
 * @return javax.swing.JPanel
386 */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
388 private javax.swing.JPanel getDisplayPage() {
   if (ivjDisplayPage == null) {
390   try {
         ivjDisplayPage = new javax.swing.JPanel();
392      ivjDisplayPage.setName("DisplayPage");
         ivjDisplayPage.setLayout(new java.awt.GridBagLayout());
394
         java.awt.GridBagConstraints constraintsDisplayQualityLabel = new java.awt.GridBagConstraints();
396      constraintsDisplayQualityLabel.gridx = 0; constraintsDisplayQualityLabel.gridy = 0;
         constraintsDisplayQualityLabel.gridwidth = 2;
398      constraintsDisplayQualityLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsDisplayQualityLabel.weighty = 1.0;
400      constraintsDisplayQualityLabel.insets = new java.awt.Insets(6, 6, 6, 6);
         getDisplayPage().add(getDisplayQualityLabel(), constraintsDisplayQualityLabel);
402
         java.awt.GridBagConstraints constraintsLayoutSchemeLabel = new java.awt.GridBagConstraints();
404      constraintsLayoutSchemeLabel.gridx = 0; constraintsLayoutSchemeLabel.gridy = 1;
         constraintsLayoutSchemeLabel.gridwidth = 2;
406      constraintsLayoutSchemeLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
         constraintsLayoutSchemeLabel.weighty = 1.0;
408      constraintsLayoutSchemeLabel.insets = new java.awt.Insets(6, 6, 6, 6);
         getDisplayPage().add(getLayoutSchemeLabel(), constraintsLayoutSchemeLabel);
410
         java.awt.GridBagConstraints constraintsDisplayQualityBox = new java.awt.GridBagConstraints();
412      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       }
```

```
      }
678   return ivjLayoutSchemeBox;
   }
680 /**
    * Return the TextColourLabel property value.
682  * @return javax.swing.JLabel
    */
684 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getLayoutSchemeLabel() {
686   if (ivjLayoutSchemeLabel == null) {
         try {
688         ivjLayoutSchemeLabel = new javax.swing.JLabel();
            ivjLayoutSchemeLabel.setName("LayoutSchemeLabel");
690         ivjLayoutSchemeLabel.setText("Layout␣scheme:");
            // user code begin {1}
692         // user code end
         } catch (java.lang.Throwable ivjExc) {
694         // user code begin {2}
            // user code end
696         handleException(ivjExc);
         }
698   }
      return ivjLayoutSchemeLabel;
700 }
   /**
702  * Return the NameLabel property value.
    * @return javax.swing.JLabel
704  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
706 private javax.swing.JLabel getNameLabel() {
      if (ivjNameLabel == null) {
708      try {
            ivjNameLabel = new javax.swing.JLabel();
710         ivjNameLabel.setName("NameLabel");
            ivjNameLabel.setText("Name:");
712         // user code begin {1}
            // user code end
714      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
716         // user code end
            handleException(ivjExc);
718      }
      }
720   return ivjNameLabel;
   }
722 /**
    * Return the JTextField1 property value.
724  * @return javax.swing.JTextField
    */
726 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JTextField getNameTextField() {
728   if (ivjNameTextField == null) {
         try {
730         ivjNameTextField = new javax.swing.JTextField();
            ivjNameTextField.setName("NameTextField");
732         // user code begin {1}
            // user code end
734      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
736         // user code end
            handleException(ivjExc);
738      }
      }
740   return ivjNameTextField;
   }
742 /**
```

```
     * 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}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjPersonalPage;
   }
   /**
    * Return the ServerName property value.
    * @return javax.swing.JLabel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getServerName() {
      if (ivjServerName == null) {
         try {
            ivjServerName = new javax.swing.JLabel();
            ivjServerName.setName("ServerName");
            ivjServerName.setText("Server␣name:");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjServerName;
   }
   /**
    * Return the JTextField1 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 ServerPage property value.
    * @return javax.swing.JPanel
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JPanel getServerPage() {
      if (ivjServerPage == null) {
         try {
            ivjServerPage = new javax.swing.JPanel();
            ivjServerPage.setName("ServerPage");
            ivjServerPage.setOpaque(true);
            ivjServerPage.setLayout(new java.awt.GridBagLayout());

            java.awt.GridBagConstraints constraintsServerName = new java.awt.GridBagConstraints();
            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() );
```

```
1202    }
        if ( getLayoutSchemeBox().getSelectedItem() != null )
1204    {
           p.setProperty( ClientConstants.getLAYOUT_SCHEME(), (String)getLayoutSchemeBox().getSelectedItem()
                ();
1206    }
        p.setProperty( ClientConstants.getAUTHOR_NAME(), (String)getNameTextField().getText() );
1208    p.setProperty( ClientConstants.getSIGNATURE(), (String)getSignaturePane().getText() );
        p.setProperty( ClientConstants.getSERVER_NAME(), (String)getServerNameField().getText() );
1210    p.setProperty( ClientConstants.getSERVER_PORT(), (String)getServerPortField().getText() );
        if ( getConnectionTypeBox().getSelectedItem() != null )
1212    {
           p.setProperty( ClientConstants.getCONNECTION_TYPE(), (String)getConnectionTypeBox().
                getSelectedItem() );
1214    }
        p.setProperty( ClientConstants.getUPDATE_INTERVAL(), (String)getUpdateIntervalField().getText() );
1216    // store the inverse of the slider value to have it interpreted as a temporal value
        p.setProperty(  ClientConstants.getANIMATION_DURATION_IN_MILLISECS(),
1218              String.valueOf( ClientConstants.getANIMATION_MAX_SPEED() - getAnimationSpeedSlider().
                     getValue() ));

1220    ClientInfo.setPreferences( p );
        ClientInfo.writePreferences();
1222    dispose();
    }
1224 }
```

Listing C.19: PZCamera.java

```
    package clientapp;
 2
    /**
 4   * 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.
 6  * Creation date: (01-08-00 16:49:47)
    * @author:
 8  */
    public class PZCamera extends edu.umd.cs.jazz.ZCamera {
10    private boolean centering = false;
    /**
12  * PZCamera constructor comment.
    */
14  public PZCamera() {
      super();
16  }
    /**
18  * PZCamera constructor comment.
    * @param layer edu.umd.cs.jazz.ZLayerGroup
20  * @param aSurface edu.umd.cs.jazz.ZDrawingSurface
    */
22  public PZCamera(edu.umd.cs.jazz.ZLayerGroup layer, edu.umd.cs.jazz.ZDrawingSurface aSurface) {
      super(layer, aSurface);
24  }
    /**
26  * Insert the method's description here.
    * Creation date: (01-08-00 16:54:15)
28  * @return boolean
    */
30  public boolean isCentering() {
      return centering;
32  }
    /**
34  * Insert the method's description here.
    * Creation date: (01-08-00 16:56:11)
36  * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
    */
```

```
38  public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
    {
40     if ( isCentering() == true )
       {
42        return;
       }
44  }
    /**
46   * Insert the method's description here.
     * Creation date: (01-08-00 16:54:15)
48   * @param newCentering boolean
     */
50  public void setCentering(boolean newCentering) {
       centering = newCentering;
52  }
    }
```

Listing C.20: PZCameraListener.java

```
    package clientapp;
 2
    /**
 4   * Simple extension of the JAZZ camera adapter.
     * Creation date: (13-07-00 12:16:07)
 6   * @author:
     */
 8  public class PZCameraListener extends edu.umd.cs.jazz.event.ZCameraAdapter
    {
10  /**
     * Insert the method's description here.
12   * Creation date: (13-07-00 12:17:43)
     * @param cameraEvent edu.umd.cs.jazz.event.ZCameraEvent
14   */
    public void viewChanged(edu.umd.cs.jazz.event.ZCameraEvent cameraEvent)
16  {

18  }
    }
```

Listing C.21: PZoomHandler.java

```
    package clientapp;
 2
    /**
 4   * 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)
 6   * @author:
     */
 8  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
10     // sker hvis der er flere instanser af klienten
       private static boolean Zooming = false;
12     private static java.awt.Rectangle targetRectangle = null;
    /**
14   * PZoomHandler constructor comment.
     * @param node edu.umd.cs.jazz.ZNode
16   */
    public PZoomHandler(edu.umd.cs.jazz.ZNode node) {
18     super(node);
    }
20  /**
     * Insert the method's description here.
22   * 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);
16  }
    /**
18   * Insert the method's description here.
     * Creation date: (17-06-00 13:45:49)
20   * @return boolean
     */
22  public static boolean isPanning() {
       return Panning;
24  }
    /**
26   * Insert the method's description here.
     * Creation date: (17-06-00 13:47:43)
28   * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
     */
30  public void mousePressed(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent)
    {
32     Panning = true;
       super.mousePressed(mouseEvent);
34  }
    /**
36   * Insert the method's description here.
     * Creation date: (17-06-00 13:48:07)
38   * @param mouseEvent edu.umd.cs.jazz.event.ZMouseEvent
     */
40  public void mouseReleased(edu.umd.cs.jazz.event.ZMouseEvent mouseEvent)
    {
42     Panning = false;
       super.mouseReleased(mouseEvent);
44  }
    }
```

Listing C.23: PZSwing.java

```
    package clientapp;
 2
    /**
 4   * Customization of the JAZZ Swing component wrapper.
     * Creation date: (17-06-00 10:17:04)
 6   * @author:
     */
 8  public class PZSwing extends edu.umd.cs.jazz.component.ZSwing
    {
10     private edu.umd.cs.jazz.ZVisualLeaf zoomingNode = null;
       private float scaleFactor = (float)1.0;
12     private java.awt.geom.AffineTransform originalTransform = null;
       private static boolean updating = false;
14  /**
     * PZSwing constructor comment.
16   * @param zbc edu.umd.cs.jazz.util.ZCanvas
     * @param component javax.swing.JComponent
18   */
    public PZSwing(edu.umd.cs.jazz.util.ZCanvas zbc, javax.swing.JComponent component) {
20     super(zbc, component);
    }
22  /**
     * Insert the method's description here.
24   * Creation date: (30-07-00 17:50:17)
     * @return java.awt.geom.AffineTransform
26   */
    public java.awt.geom.AffineTransform getOriginalTransform() {
28     return originalTransform;
    }
30  /**
     * Insert the method's description here.
32   * 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.
100  * Creation date: (30-07-00 17:50:17)
     * @param newOriginalTransform java.awt.geom.AffineTransform
102  */
    public void setOriginalTransform(java.awt.geom.AffineTransform newOriginalTransform) {
104    originalTransform = newOriginalTransform;
    }
106 /**
     * Insert the method's description here.
108  * Creation date: (30-07-00 17:43:52)
     * @param newScaleFactor float
110  */
    public void setScaleFactor(float newScaleFactor) {
112    scaleFactor = newScaleFactor;
    }
114 /**
     * Insert the method's description here.
116  * Creation date: (07-09-00 23:31:48)
     * @param newUpdating boolean
118  */
    public static void setUpdating(boolean newUpdating) {
120    updating = newUpdating;
    }
122 /**
     * Insert the method's description here.
124  * Creation date: (30-07-00 17:26:06)
     * @param newZoomingNode edu.umd.cs.jazz.ZVisualLeaf
126  */
    public void setZoomingNode(edu.umd.cs.jazz.ZVisualLeaf newZoomingNode) {
128    zoomingNode = newZoomingNode;
    }
130 }
```

Listing C.24: PZText.java

```
    package clientapp;
 2
    /**
 4   * Component for representing text items in the client panorama
     * Creation date: (16-06-00 23:05:35)
 6   * @author:
     */
 8  public class PZText extends edu.umd.cs.jazz.component.ZText
    {
10
    /**
12   * PZText constructor comment.
     */
14  public PZText() {
       super();
16  }
    /**
18   * PZText constructor comment.
     * @param str java.lang.String
20   */
    public PZText(String str) {
22     super(str);
    }
24  /**
     * PZText constructor comment.
26   * @param str java.lang.String
     * @param font java.awt.Font
28   */
    public PZText(String str, java.awt.Font font) {
30     super(str, font);
    }
32  /**
```

```
     * Insert the method's description here.
34   * Creation date: (16-06-00 23:07:51)
     * @param renderContext edu.umd.cs.jazz.util.ZRenderContext
36   */
    public void render(edu.umd.cs.jazz.util.ZRenderContext renderContext)
38  {
       //renderContext.setGreekText(true);
40     super.render(renderContext);
    }
42  }
```

Listing C.25: RemoveFiles.java

```
    package clientapp;
 2

 4   import java.awt.event.*;
     import javax.swing.border.*;
 6   import javax.swing.BorderFactory;
     import javax.swing.DefaultListModel;
 8   import java.io.File;

10  /**
     * This class implements the remove files dialog box.
12   * Creation date: (22-06-00 23:00:55)
     * @author:
14   */

16  public class RemoveFiles extends javax.swing.JDialog {

18     public class RemoveButtonHandler implements ActionListener
       {
20        public void actionPerformed(ActionEvent actionEvent)
          {
22           submitAndClose();
          }
24     }

26     public class CancelButtonHandler implements ActionListener
       {
28        public void actionPerformed(ActionEvent actionEvent)
          {
30           dispose();
          }
32     }

34     public class MoveLeftButtonHandler implements ActionListener
       {
36        public void actionPerformed(ActionEvent actionEvent)
          {
38           moveFromDisplayedToRemove();
          }
40     }

42     public class MoveRightButtonHandler implements ActionListener
       {
44        public void actionPerformed(ActionEvent actionEvent)
          {
46           moveFromRemoveToDisplayed();
          }
48     }
       private javax.swing.JButton ivjCancelButton = null;
50     private javax.swing.JPanel ivjJDialogContentPane = null;
       private javax.swing.JScrollPane ivjJScrollPane1 = null;
52     private javax.swing.JButton ivjRemoveButton = null;
       private javax.swing.DefaultListModel listModel = new DefaultListModel();
54     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
```

```
     */
386  public javax.swing.DefaultListModel getListModel() {
       return listModel;
388  }
     /**
390   * Return the JPanel2 property value.
      * @return javax.swing.JPanel
392   */
     /* WARNING: THIS METHOD WILL BE REGENERATED. */
394  private javax.swing.JPanel getMainPanel() {
       if (ivjMainPanel == null) {
396      try {
           ivjMainPanel = new javax.swing.JPanel();
398        ivjMainPanel.setName("MainPanel");
           ivjMainPanel.setLayout(new java.awt.GridBagLayout());
400
           java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
402        constraintsButtonPanel.gridx = 1; constraintsButtonPanel.gridy = 1;
           constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
404        getMainPanel().add(getButtonPanel(), constraintsButtonPanel);

406        java.awt.GridBagConstraints constraintsDisplayedLabel = new java.awt.GridBagConstraints();
           constraintsDisplayedLabel.gridx = 0; constraintsDisplayedLabel.gridy = 0;
408        constraintsDisplayedLabel.anchor = java.awt.GridBagConstraints.WEST;
           constraintsDisplayedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
410        getMainPanel().add(getDisplayedLabel(), constraintsDisplayedLabel);

412        java.awt.GridBagConstraints constraintsJScrollPane1 = new java.awt.GridBagConstraints();
           constraintsJScrollPane1.gridx = 0; constraintsJScrollPane1.gridy = 1;
414        constraintsJScrollPane1.fill = java.awt.GridBagConstraints.BOTH;
           constraintsJScrollPane1.anchor = java.awt.GridBagConstraints.WEST;
416        constraintsJScrollPane1.weightx = 1.0;
           constraintsJScrollPane1.weighty = 1.0;
418        constraintsJScrollPane1.insets = new java.awt.Insets(4, 4, 4, 4);
           getMainPanel().add(getJScrollPane1(), constraintsJScrollPane1);
420
           java.awt.GridBagConstraints constraintsJScrollPane2 = new java.awt.GridBagConstraints();
422        constraintsJScrollPane2.gridx = 2; constraintsJScrollPane2.gridy = 1;
           constraintsJScrollPane2.fill = java.awt.GridBagConstraints.BOTH;
424        constraintsJScrollPane2.anchor = java.awt.GridBagConstraints.EAST;
           constraintsJScrollPane2.weightx = 1.0;
426        constraintsJScrollPane2.weighty = 1.0;
           constraintsJScrollPane2.insets = new java.awt.Insets(4, 4, 4, 4);
428        getMainPanel().add(getJScrollPane2(), constraintsJScrollPane2);

430        java.awt.GridBagConstraints constraintsDocumentsToRemove = new java.awt.GridBagConstraints();
           constraintsDocumentsToRemove.gridx = 2; constraintsDocumentsToRemove.gridy = 0;
432        constraintsDocumentsToRemove.anchor = java.awt.GridBagConstraints.WEST;
           constraintsDocumentsToRemove.insets = new java.awt.Insets(4, 4, 4, 4);
434        getMainPanel().add(getDocumentsToRemove(), constraintsDocumentsToRemove);
           // user code begin {1}
436        getMainPanel().setBorder( ClientConstants.createDialogBorder( getMainPanel().getBackground() ));
           // user code end
438      } catch (java.lang.Throwable ivjExc) {
           // user code begin {2}
440        // user code end
           handleException(ivjExc);
442      }
       }
444    return ivjMainPanel;
     }
446  /**
      * Return the AddButton property value.
448   * @return javax.swing.JButton
      */
450  /* 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}
            ivjRemoveButton.addActionListener(new RemoveButtonHandler() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjRemoveButton;
   }
   /**
    * Return the RemoveList property value.
    * @return javax.swing.JList
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JList getRemoveList() {
      if (ivjRemoveList == null) {
         try {
            ivjRemoveList = new javax.swing.JList();
            ivjRemoveList.setName("RemoveList");
            ivjRemoveList.setBackground(java.awt.Color.white);
            ivjRemoveList.setBounds(0, 0, 160, 120);
            // user code begin {1}
            ivjRemoveList.setModel(removeListModel);
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjRemoveList;
   }
   /**
    * Insert the method's description here.
    * Creation date: (23-06-00 19:55:23)
    * @return javax.swing.DefaultListModel
    */
   public javax.swing.DefaultListModel getRemoveListModel() {
      return removeListModel;
   }
   /**
    * 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: (22-06-00 23:11:53)
    */
   public void initialize() {
      try {
         // user code begin {1}
         // user code end
         setName("RemoveFiles");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
         setSize(600, 300);
         setTitle("Remove␣documents");
         setContentPane(getJDialogContentPane());
      } catch (java.lang.Throwable ivjExc) {
```

```
         handleException(ivjExc);
582   }
      // user code begin {2}
584   getJDialogContentPane().setBorder( javax.swing.BorderFactory.createMatteBorder(6,6,6,6,
          getJDialogContentPane().getBackground() ));
      // user code end
586   }
      /**
588    * main entrypoint - starts the part when it is run as an application
       * @param args java.lang.String[]
590    */
      public static void main(java.lang.String[] args) {
592      try {
            RemoveFiles aRemoveFiles;
594         aRemoveFiles = new RemoveFiles();
            aRemoveFiles.setModal(true);
596         aRemoveFiles.addWindowListener(new java.awt.event.WindowAdapter() {
               public void windowClosing(java.awt.event.WindowEvent e) {
598               System.exit(0);
               };
600         });
            aRemoveFiles.setVisible(true);
602      } catch (Throwable exception) {
            System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
604         exception.printStackTrace(System.out);
         }
606   }
      /**
608    * Insert the method's description here.
       * Creation date: (23-06-00 20:04:34)
610    */
      public void moveFromDisplayedToRemove()
612   {
         for (int i = listModel.size()-1; i >= 0; i--)
614      {
            if ( getDisplayedList().isSelectedIndex(i) )
616         {
               removeListModel.addElement( listModel.get(i) );
618            listModel.remove(i);
            }
620      }
      }
622   /**
       * Insert the method's description here.
624    * Creation date: (23-06-00 20:05:56)
       */
626   public void moveFromRemoveToDisplayed()
      {
628      for (int i = removeListModel.size()-1; i >= 0; i--)
         {
630         if ( getRemoveList().isSelectedIndex(i) )
            {
632            listModel.addElement( removeListModel.get(i) );
               removeListModel.remove(i);
634         }
         }
636   }
      /**
638    * Insert the method's description here.
       * Creation date: (23-06-00 00:15:50)
640    * @param newClientApplication clientapp.ClientApplication
       */
642   public void setClientApplication(ClientApplication newClientApplication) {
         clientApplication = newClientApplication;
644   }
      /**
```

```
646   * Insert the method's description here.
      * Creation date: (22-06-00 23:40:02)
648   * @param newListModel javax.swing.DefaultListModel
      */
650   public void setListModel(javax.swing.DefaultListModel newListModel) {
        listModel = newListModel;
652   }
      /**
654    * Insert the method's description here.
       * Creation date: (23-06-00 19:55:23)
656    * @param newRemoveListModel javax.swing.DefaultListModel
       */
658   public void setRemoveListModel(javax.swing.DefaultListModel newRemoveListModel) {
        removeListModel = newRemoveListModel;
660   }
      /**
662    * Insert the method's description here.
       * Creation date: (19-07-00 02:38:38)
664    */
      public void show()
666   {
        listModel.clear();
668     removeListModel.clear();
        fillListBox( clientApplication.getClientManager().getClientInfo().getLocalFiles() );
670     // default positioning puts the dialog at center of screen. The method used is a slight hack, but
            works fine.
        setLocationRelativeTo( this );
672     setLocation( (int) getLocation().getX(), (int) getLocation().getY() );
        super.show();
674   }
      /**
676    * Insert the method's description here.
       * Creation date: (23-06-00 00:12:51)
678    */
      public void submitAndClose()
680   {
        if ( removeListModel.size() > 0)
682     {
          File removables[] = new File[removeListModel.size()];
684       File remainingFiles[] = new File[listModel.size()];
          for (int i = 0; i < removables.length; i++)
686       {
            removables[i] = (File)removeListModel.get(i);
688       }
          clientApplication.getClientManager().removeDocuments( removables );
690     }
        else
692     {
          clientApplication.initMessageArea( 1, 1, 1, ClientConstants.getNOTHING_TO_REMOVE_MESSAGE() );
694     }
        dispose();
696   }
      }
```

Listing C.26: Validator.java

```
      package clientapp;
 2
      /**
 4     * Factory class with various static member functions for validating data entered into the client
            application.
       * Creation date: (26-07-00 15:44:56)
 6     * @author:
       */
 8    public class Validator
      {
10      private ClientApplication ClientApplication = null;
```

```
   /**
12  * Validator constructor comment.
    */
14 public Validator( ClientApplication clientApplication )
   {
16    super();
      setClientApplication( clientApplication );
18 }
   /**
20  * Insert the method's description here.
    * Creation date: (26-07-00 16:07:18)
22  * @return clientapp.ClientApplication
    */
24 public ClientApplication getClientApplication() {
      return ClientApplication;
26 }
   /**
28  * Insert the method's description here.
    * Creation date: (26-07-00 16:07:18)
30  * @param newClientApplication clientapp.ClientApplication
    */
32 public void setClientApplication(ClientApplication newClientApplication) {
      ClientApplication = newClientApplication;
34 }
   }
```

Listing C.27: GroupManagementServer.java

```
   package serverapp;
 2
   import com.sun.media.jsdt.*;
 4 /**
    * This class is intended for use by PeerView servers connecting to a JSDT session.
 6  * Creation date: (03-07-00 12:17:46)
    * @author:
 8  */
   public class GroupManagementServer implements Client, ChannelConsumer {
10    private java.lang.String name = "";
   /**
12  * GroupManagementServer constructor comment.
    */
14 public GroupManagementServer() {
      super();
16 }
   /**
18  * Insert the method's description here.
    * Creation date: (03-07-00 12:30:58)
20  * @param ai com.sun.media.jsdt.AuthenticationInfo
    */
22 public Object authenticate(AuthenticationInfo ai)
   {
24    return null;
   }
26 /**
    * dataReceived method comment.
28  */
   public synchronized void dataReceived(Data data)
30 {

32
   }
34 /**
    * Insert the method's description here.
36  * Creation date: (03-07-00 12:21:42)
    * @return java.lang.String
38  */
   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();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Insert the method's description here.
 * Creation date: (07-07-00 01:53:28)
 */
protected void finalize() throws Throwable
{
   terminateAndClose();
   super.finalize();
}
/**
 * Return the AboutDialog property value.
 * @return clientapp.AboutDialog
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private peerviewmisc.AboutDialog getAboutDialog() {
   if (ivjAboutDialog == null) {
      try {
         ivjAboutDialog = new peerviewmisc.AboutDialog();
         ivjAboutDialog.setName("AboutDialog");
         ivjAboutDialog.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 ivjAboutDialog;
}
/**
 * Return the AboutMenuItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JMenuItem getAboutMenuItem() {
   if (ivjAboutMenuItem == null) {
      try {
         ivjAboutMenuItem = new javax.swing.JMenuItem();
         ivjAboutMenuItem.setName("AboutMenuItem");
         ivjAboutMenuItem.setText("About");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   }
   return ivjAboutMenuItem;
}
/**
 * Return the ContentsMenuItem property value.
 * @return javax.swing.JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
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
```

```
384   */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
386   private javax.swing.JButton getHelpButton() {
         if (ivjHelpButton == null) {
388         try {
               ivjHelpButton = new javax.swing.JButton();
390            ivjHelpButton.setName("HelpButton");
               ivjHelpButton.setToolTipText("Help");
392            ivjHelpButton.setText("");
               ivjHelpButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
394            ivjHelpButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
               ivjHelpButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                    general/Help24.gif")));
396            ivjHelpButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
               // user code begin {1}
398            // user code end
            } catch (java.lang.Throwable ivjExc) {
400            // user code begin {2}
               // user code end
402            handleException(ivjExc);
            }
404      }
         return ivjHelpButton;
406   }
      /**
408    * Return the HelpDialog1 property value.
       * @return peerviewmisc.HelpDialog
410    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
412   private peerviewmisc.HelpDialog getHelpDialog1() {
         if (ivjHelpDialog1 == null) {
414         try {
               ivjHelpDialog1 = new peerviewmisc.HelpDialog();
416            ivjHelpDialog1.setName("HelpDialog1");
               ivjHelpDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
418            // user code begin {1}
               // user code end
420         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
422            // user code end
               handleException(ivjExc);
424         }
         }
426      return ivjHelpDialog1;
      }
428   /**
       * Return the HelpMenu property value.
430    * @return javax.swing.JMenu
       */
432   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JMenu getHelpMenu() {
434      if (ivjHelpMenu == null) {
            try {
436            ivjHelpMenu = new javax.swing.JMenu();
               ivjHelpMenu.setName("HelpMenu");
438            ivjHelpMenu.setText("Help");
               ivjHelpMenu.add(getContentsMenuItem());
440            ivjHelpMenu.add(getAboutMenuItem());
               // user code begin {1}
442            // user code end
            } catch (java.lang.Throwable ivjExc) {
444            // user code begin {2}
               // user code end
446            handleException(ivjExc);
            }
448      }
```

```
      return ivjHelpMenu;
450   }
      /**
452    * Return the JFrameContentPane property value.
       * @return javax.swing.JPanel
454    */
      /* WARNING: THIS METHOD WILL BE REGENERATED. */
456   private javax.swing.JPanel getJFrameContentPane() {
         if (ivjJFrameContentPane == null) {
458         try {
               ivjJFrameContentPane = new javax.swing.JPanel();
460            ivjJFrameContentPane.setName("JFrameContentPane");
               ivjJFrameContentPane.setLayout(new java.awt.BorderLayout());
462            getJFrameContentPane().add(getJToolBar1(), "North");
               getJFrameContentPane().add(getJPanel1(), "Center");
464            // user code begin {1}
               // user code end
466         } catch (java.lang.Throwable ivjExc) {
               // user code begin {2}
468            // user code end
               handleException(ivjExc);
470         }
         }
472      return ivjJFrameContentPane;
      }
474   /**
       * Return the JPanel1 property value.
476    * @return javax.swing.JPanel
       */
478   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JPanel getJPanel1() {
480      if (ivjJPanel1 == null) {
            try {
482            ivjJPanel1 = new javax.swing.JPanel();
               ivjJPanel1.setName("JPanel1");
484            ivjJPanel1.setLayout(new java.awt.BorderLayout());
               getJPanel1().add(getJScrollPane1(), "Center");
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 ivjJPanel1;
      }
496   /**
       * Return the JScrollPane1 property value.
498    * @return javax.swing.JScrollPane
       */
500   /* WARNING: THIS METHOD WILL BE REGENERATED. */
      private javax.swing.JScrollPane getJScrollPane1() {
502      if (ivjJScrollPane1 == null) {
            try {
504            ivjJScrollPane1 = new javax.swing.JScrollPane();
               ivjJScrollPane1.setName("JScrollPane1");
506            getJScrollPane1().setViewportView(getScrollPaneTable());
               // user code begin {1}
508            // user code end
            } catch (java.lang.Throwable ivjExc) {
510            // user code begin {2}
               // user code end
512            handleException(ivjExc);
            }
514      }
```

```
      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      }
```

```
      }
      return ivjServerMenu;
   }
   /**
    * Return the ServerSetup property value.
    * @return serverapp.ServerSetup
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private ServerSetup getServerSetup() {
      if (ivjServerSetup == null) {
         try {
            ivjServerSetup = new serverapp.ServerSetup();
            ivjServerSetup.setName("ServerSetup");
            ivjServerSetup.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 ivjServerSetup;
   }
   /**
    * Return the SetupButton property value.
    * @return javax.swing.JButton
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JButton getSetupButton() {
      if (ivjSetupButton == null) {
         try {
            ivjSetupButton = new javax.swing.JButton();
            ivjSetupButton.setName("SetupButton");
            ivjSetupButton.setToolTipText("Server␣setup");
            ivjSetupButton.setText("");
            ivjSetupButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            ivjSetupButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
            ivjSetupButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
                general/Properties24.gif")));
            ivjSetupButton.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 ivjSetupButton;
   }
   /**
    * Return the SetupMenuItem property value.
    * @return javax.swing.JMenuItem
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JMenuItem getSetupMenuItem() {
      if (ivjSetupMenuItem == null) {
         try {
            ivjSetupMenuItem = new javax.swing.JMenuItem();
            ivjSetupMenuItem.setName("SetupMenuItem");
            ivjSetupMenuItem.setText("Setup");
            // user code begin {1}
            // user code end
         } 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++ )
      {
```

```
908       tableModel.removeRow( 0 );
      }
910    tableModel.setNumRows( 0 );
       // update with contents of group directory
912    Collection values = getServerManager().getServerInfo().getGroups().values();
       Iterator iterator = values.iterator();
914    while ( iterator.hasNext() )
       {
916       WorkGroup workGroup = (WorkGroup) iterator.next();
          Vector v = workGroup.getGroupAsVector();
918       tableModel.addRow( v );
       }
920    // Distribute surplus space evenly
       getScrollPaneTable().sizeColumnsToFit( -1 );
922    getScrollPaneTable().revalidate();
       getScrollPaneTable().repaint();
924    getJScrollPane1().revalidate();
       getJScrollPane1().repaint();
926 }
    }
```

Listing C.30: ServerConstants.java

```
    package serverapp;
 2
     import javax.swing.border.*;
 4   import javax.swing.BorderFactory.*;
     import peerviewmisc.SharedConstants;
 6   import java.util.Properties;

 8   /**
     * This class holds constants specific to the PeerView server as well as those inherited from {@link #
         peerviewmisc.SharedConstants SharedConstants}.
10   *
     * Creation date: (27-06-00 20:58:01)
12   * @author:
     */
14
    public class ServerConstants extends SharedConstants
16  {
       private final static java.lang.String PREFERENCES_FILE_NAME = "sprefs.ini";
18     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";
20     private final static java.lang.String ROOT_NODE_MESSAGE_TITLE = "System␣notice";
       private final static java.lang.String PROPERTIES_HEADER = "";
22     private final static java.lang.String FRAME_HORIZONTAL_SIZE = "Frame␣horizontal␣size";
       private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame␣vertical␣size";
24     private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";
       private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";
26     private final static java.lang.String FRAME_X_COORDINATE = "Frame␣x␣coordinate";
       private final static java.lang.String FRAME_Y_COORDINATE = "Frame␣y␣coordinate";
28     private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";
       private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";
30     final static String DEFAULT_PREFERENCES[][] = { {SERVER_NAME, DEFAULT_SERVER_NAME},
                                       {SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT)},
32                                     {CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE},
                                       {FRAME_HORIZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE},
34                                     {FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE},
                                       {FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE},
36                                     {FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE},
                                       {getREGISTRY_PORT(), getDEFAULT_REGISTRY_PORT()},
38                                    };

40     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;
```

```
102 }
    /**
104  * Insert the method's description here.
     * Creation date: (05-08-00 19:50:26)
106  * @return java.lang.String
     */
108 public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
       return DEFAULT_FRAME_VERTICAL_SIZE;
110 }
    /**
112  * Insert the method's description here.
     * Creation date: (05-08-00 19:51:49)
114  * @return java.lang.String
     */
116 public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
       return DEFAULT_FRAME_X_COORDINATE;
118 }
    /**
120  * Insert the method's description here.
     * Creation date: (05-08-00 19:52:26)
122  * @return java.lang.String
     */
124 public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
       return DEFAULT_FRAME_Y_COORDINATE;
126 }
    /**
128  * Insert the method's description here.
     * Creation date: (03-07-00 14:34:27)
130  * @return java.util.Properties
     */
132 public static java.util.Properties getDEFAULT_PREFERENCES()
    {
134    Properties p = new Properties();

136    for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
       {
138       p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
       }
140    return p;
    }
142 /**
     * Insert the method's description here.
144  * Creation date: (05-08-00 19:49:06)
     * @return java.lang.String
146  */
    public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
148    return FRAME_HORIZONTAL_SIZE;
    }
150 /**
     * Insert the method's description here.
152  * Creation date: (05-08-00 20:05:17)
     * @return int
154  */
    public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
156    return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
    }
158 /**
     * Insert the method's description here.
160  * Creation date: (05-08-00 19:49:24)
     * @return java.lang.String
162  */
    public final static java.lang.String getFRAME_VERTICAL_SIZE() {
164    return FRAME_VERTICAL_SIZE;
    }
166 /**
     * 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
```

```
234   */
      public final static java.lang.String getPROPERTIES_HEADER() {
236     return PROPERTIES_HEADER;
      }
238   /**
       * Insert the method's description here.
240    * Creation date: (10-07-00 00:50:05)
       * @param documentName java.lang.String
242    * @param senderName java.lang.String
       */
244
      public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
          Date creationDate)
246   {
        return "This␣is␣the␣discussion␣forum␣for␣the␣document␣" + documentName + "␣which␣was␣added␣by␣" +
            senderName +
248           "␣on␣" + creationDate.toString();
      }
250   /**
       * Insert the method's description here.
252    * Creation date: (10-07-00 00:50:05)
       * @param documentName java.lang.String
254    * @param senderName java.lang.String
       */
256
      public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
          GregorianCalendar creationDate)
258   {
        return "This␣is␣the␣discussion␣forum␣for␣the␣document␣" + documentName + "␣which␣was␣added␣by␣" +
            senderName +
260           "␣on␣" + creationDate.getTime().toString();
      }
262   /**
       * Insert the method's description here.
264    * Creation date: (10-07-00 00:48:37)
       * @return java.lang.String
266    */
      public final static java.lang.String getROOT_NODE_MESSAGE_TITLE() {
268     return ROOT_NODE_MESSAGE_TITLE;
      }
270   }
```

Listing C.31: ServerInfo.java

```
      package serverapp;
 2
       import java.util.Properties;
 4     import java.util.Hashtable;
       import peerviewmisc.*;
 6     import java.io.*;

 8     /**
       * The PeerView server main entity object. This class holds data structures and fields shared by several
            server components.
10     * Creation date: (28-06-00 22:16:34)
       * @author:
12     */

14    public class ServerInfo
      {
16      private com.sun.media.jsdt.Channel groupManagementChannel = null;
        private java.util.Hashtable groups = new Hashtable();
18      private static java.util.Properties preferences = null;
        private com.sun.media.jsdt.Session groupManagementSession;
20      private int mostRecentPortNumber = ServerConstants.getDEFAULT_SERVER_PORT();
        private java.util.Hashtable discussions = new Hashtable();
22      private java.util.Hashtable messages = new Hashtable();
```

```
   private static boolean visible = true;
/**
 * ServerInfo constructor comment.
 */
public ServerInfo()
{
   super();
}
/**
 * Insert the method's description here.
 * Creation date: (14-08-00 23:05:59)
 * @exception java.lang.Throwable The exception description.
 */
protected void finalize() throws java.lang.Throwable
{
}
/**
 * Insert the method's description here.
 * Creation date: (09-07-00 15:03:03)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getDiscussions() {
   return discussions;
}
/**
 * Insert the method's description here.
 * Creation date: (01-07-00 01:20:55)
 * @return com.sun.media.jsdt.Channel
 */
public com.sun.media.jsdt.Channel getGroupManagementChannel() {
   return groupManagementChannel;
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 17:12:12)
 * @return com.sun.media.jsdt.Session
 */
public com.sun.media.jsdt.Session getGroupManagementSession() {
   return groupManagementSession;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:11:40)
 * @return java.util.Properties
 */
public java.util.Hashtable getGroups()
{
   if ( groups == null )
   {
      groups = new Hashtable();
   }
   return groups;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 01:27:49)
 * @return java.util.Hashtable
 */
public java.util.Hashtable getMessages() {
   return messages;
}
/**
 * Insert the method's description here.
 * Creation date: (07-07-00 17:30:38)
 * @return int
 */
```

```
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
218  */
    public static void writePreferences()
220 {
       // write Properties object to disk
222    try
       {
224       FileOutputStream f = new FileOutputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
          getPreferences().store( f, ServerConstants.getPROPERTIES_HEADER() );
226       f.close();
       }
228    catch (IOException ie)
       {
230       System.err.println( ie.toString() );
          System.exit(0);
232    }

234 }
    }
```

Listing C.32: ServerManager.java

```
    package serverapp;
 2
     import com.sun.media.jsdt.*;
 4   import java.io.*;
     import java.util.Hashtable;
 6   import peerviewmisc.*;
     import java.util.Collection;
 8   import java.util.Iterator;
     import javax.swing.tree.*;
10   import java.util.Date;
     import com.sun.media.jsdt.event.*;
12
    /**
14   * This is the main control class for the PeerView server application. It contains methods and control
         structures shared
     * by several server components.
16   * Creation date: (27-06-00 20:17:35)
     * @author:
18   */
    public class ServerManager
20  {
       private ServerInfo serverInfo = null;
22
       public class DocNameAndSize implements Serializable
24     {
          public String docName = null;
26        public long size = 0;

28        public DocNameAndSize( String docNameArg, long sizeArg )
          {
30           docName = docNameArg;
             size = sizeArg;
32        }
          public String getDocName()
34        {
             return docName;
36        }
          public void setDocName( String newDocName )
38        {
             docName = newDocName;
40        }
          public long getSize()
42        {
             return size;
44        }
```

```
      public void setSize( long newSize )
46    {
         size = newSize;
48    }
   };
50
   public class GroupListener extends SessionAdaptor
52 {
      public String workGroupID = null;
54    public GroupListener( String workGroupIDArg )
      {
56       workGroupID = workGroupIDArg;
      }
58
      public void sessionJoined( SessionEvent se )
60    {
         clientJoinedGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID
              ));
62    }

64    public void sessionLeft( SessionEvent se )
      {
66       System.out.println( "Exit client: " + se.getClientName() );
         try
68       {
         for ( int i = 0 ; i < se.getSession().listClientNames().length; i++ )
70       {
            System.out.println( se.getSession().listClientNames()[i] );
72       }
         }
74       catch ( Exception E )
         {
76       }
         clientLeftGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID ))
              ;
78    }
   }
80
   public class DefaultClient implements com.sun.media.jsdt.Client
82 {
      String name;
84
      public DefaultClient (String nameArg )
86    {
         name = nameArg;
88    }
      public Object authenticate ( com.sun.media.jsdt.AuthenticationInfo ai )
90    {
         return null;
92    }
      public String getName ()
94    {
         return name;
96    }
   };
98
   public class GroupManagementServer implements Client, ChannelConsumer
100 {
      String name;
102
      public GroupManagementServer()
104   {
      }
106
      public String getName()
108   {
```

```
      return name;
   }

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

   public Object authenticate ( AuthenticationInfo ai )
   {
      return null;
   }

   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);
      }
      System.out.println( "Server␣fik␣" + incomingPackage.getProperties().getProperty( DataPackage.
          getDESCRIPTION() ));
      // Determine the type of incoming package
      if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getREQUEST_GROUP_DIRECTORY() ))
      {
         sendGroupDirectory ( data.getSenderName() );
      }

      else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getREQUEST_ADD_GROUP() ))
      {
         addGroup( (peerviewmisc.WorkGroup) incomingPackage.getContents() );
      }

      else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getCLIENT_ADDED_DOCUMENTS() ))
      {
         createDiscussionForum( (String) incomingPackage.getProperties().getProperty( DataPackage.
             getAUTHOR_NAME() ), (String) incomingPackage.getContents(), data.getSenderName() );
      }

      else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getREQUEST_TOPIC_TREE() ))
      {
         sendTopicTree( (String) incomingPackage.getContents(), data.getSenderName() );
      }

      else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getREQUEST_MESSAGE() ))
      {
         sendMessage( (String) incomingPackage.getProperties().getProperty( DataPackage.getMESSAGE_ID
             () ), data.getSenderName() );
      }

      else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
          equalsIgnoreCase( ServerConstants.getCLIENT_ADDED_MESSAGE() ))
      {
         addMessage( (Message) incomingPackage.getContents() );
      }
```

```
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
```

```
     */
220  public void activateGroup(WorkGroup workGroup)
     {
222        Session session = null;
           URLString url = null;
224        GroupManagementServer newServer = new GroupManagementServer();
           newServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME() ));
226
           try
228        {
           boolean created = false;
230  /*    // Cycle through port numbers until an unoccupied one is found and use it for creating a new
          session
           while ( created == false )
232           {
              try
234           {
     */       url = URLString.createSessionURL( ServerInfo.getPreferences().getProperty ( ServerConstants.
          getSERVER_NAME() ),
236                                        Integer.parseInt( ServerInfo.getPreferences().getProperty(
                                               ServerConstants.getSERVER_PORT() )),
                                           ServerInfo.getPreferences().getProperty ( ServerConstants.
                                               getCONNECTION_TYPE() ),
238                                        workGroup.getWorkGroupID() );
              // System.out.println( workGroup.getWorkGroupID() );
240           session = SessionFactory.createSession( newServer, url, true );
              created = true;
242           session.addSessionListener( new GroupListener( workGroup.getWorkGroupID() ));
     /*    }
244           catch ( PortInUseException PIUE )
           {
246           serverInfo.setMostRecentPortNumber( serverInfo.getMostRecentPortNumber() + 1 );
           }
248
           }
250  */    Channel groupChannel = session.createChannel( newServer, workGroup.getWorkGroupChannelID(), true,
          true, true );
           groupChannel.addConsumer( newServer, newServer );
252        workGroup.setSessionURL( url );
           }
254        catch ( JSDTException JSDTE )
           {
256           handleJSDTException( JSDTE );
           }
258
     }
260  /**
      * Activate the groups read from disk file by creating and registering the appropriate sessions and
          channels.
262   * Creation date: (05-07-00 23:33:46)
      */
264  public void activateGroups()
     {
266     Collection values = serverInfo.getGroups().values();

268     Iterator iterator = values.iterator();
        while ( iterator.hasNext() )
270     {
           activateGroup( (WorkGroup) iterator.next() );
272
        }
274  }
     /**
276   * Insert the method's description here.
      * Creation date: (03-07-00 19:05:53)
278   * @param workGroup peerviewmisc.WorkGroup
```

```
     */
280  public void addGroup(peerviewmisc.WorkGroup workGroup)
     {
282     serverInfo.getGroups().put( workGroup.getWorkGroupID(), workGroup);
        activateGroup( workGroup );
284     broadcastGroupDirectory();
        serverApplication.updateTable();
286  }
     /**
288   * Insert the method's description here.
      * Creation date: (11-07-00 21:12:23)
290   * @param peerviewmisc.Message
      */
292  public void addMessage(Message newMessage)
     {
294     getServerInfo().getMessages().put( newMessage.getMessageID(), newMessage );
     }
296  /**
      * Insert the method's description here.
298   * Creation date: (20-09-00 03:03:54)
      * @param senderName java.lang.String
300   * @param documents java.util.Vector
      */
302  public void broadcastDocumentRemovals(String senderName, WorkGroup workGroup, java.util.Vector documents
         )
     {
304     DataPackage outgoingPackage = new DataPackage();
        outgoingPackage.setContents( documents );
306     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
           getREMOVE_DOCUMENTS() );
        outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), workGroup.getWorkGroupID
           () );
308     outgoingPackage.getProperties().setProperty( DataPackage.getCLIENT_NAME(), senderName );
        outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ServerInfo.getPreferences
           ().getProperty( ServerConstants.getSERVER_NAME() ));
310     try
        {
312        URLString url = workGroup.getSessionURL();
           Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
314        Channel groupChannel = session.createChannel( getGroupManagementServer(), workGroup.
              getWorkGroupChannelID(), true, true, false);
           Data data = new Data( outgoingPackage );
316        groupChannel.sendToOthers( getGroupManagementServer(), data );
        }
318     catch ( JSDTException JSDTE )
        {
320        handleJSDTException( JSDTE );
        }
322  }
     /**
324   * Insert the method's description here.
      * Creation date: (22-07-00 20:08:48)
326   * @param workGroup peerviewmisc.WorkGroup
      */
328  public void broadcastGroupDirectory()
     {
330     DataPackage outgoingPackage = new DataPackage();

332     outgoingPackage.setContents( getServerInfo().getGroups() );
        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
           getGROUP_DIRECTORY() );
334     try
        {
336        if ( getServerInfo().getGroupManagementChannel() != null )
           {
338           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
```

```
400   */
      public void clientJoinedGroup(String clientName, WorkGroup workGroup)
402   {
         WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
               );
404
         groupToBeUpdated.getParticipants().add( clientName );
406      groupToBeUpdated.getDocuments().put( clientName, new java.util.Vector() );
         getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
408      broadcastGroupDirectory();
      }
410   /**
       * Forcibly expel client from group management session if it has not already left. This to ensure
           consistency when clients
412    * terminate abruptly, i.e. do not exit properly.
       * Creation date: (01-11-00 23:45:19)
414    * @param clientName java.lang.String
       */
416   public void clientLeft(String clientName)
      {
418      try
         {
420         for ( int i = 0 ; i < getServerInfo().getGroupManagementSession().listClientNames().length; i++ )
            {
422            if ( getServerInfo().getGroupManagementSession().listClientNames()[i].equalsIgnoreCase(
                  clientName ))
               {
424                  //System.out.println( "Found client : " + clientName );
                     DefaultClient tempClient = new DefaultClient( clientName );
426                  getServerInfo().getGroupManagementSession().leave( tempClient );
                     break;
428            }
            }
430      }
         catch ( JSDTException JSDTE )
432      {
            handleJSDTException( JSDTE );
434      }
      }
436   /**
       * This method is triggered when a client leaves a group. The server updates the group's entry in the
           group directory
438    * 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.
440    * Creation date: (19-09-00 17:55:46)
       * @param clientName java.lang.String
442    * @param workGroup peerviewmisc.WorkGroup
       */
444   public void clientLeftGroup(String clientName, WorkGroup workGroup)
      {
446      WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
               );

448      if ( groupToBeUpdated.getParticipants().contains( clientName ))
         {
450         java.util.Vector documentNames = new java.util.Vector();
            java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( clientName );
452         Iterator docIterator = documents.iterator();
            while ( docIterator.hasNext() )
454         {
               Pair docNameAndSize = (Pair) docIterator.next();
456            groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt(
                     docNameAndSize.getValue() ));
               groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
458            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)
```

```
1086   */
       public void writeMessages()
1088   {
          try
1090      {
             FileOutputStream fostream = new FileOutputStream( ServerConstants.getMESSAGES_FILE_NAME() );
1092         ObjectOutputStream oostream = new ObjectOutputStream( fostream );
             oostream.writeObject( getServerInfo().getMessages() );
1094         // System.out.println( "WroteMessages" );
             oostream.flush();
1096         oostream.close();
          }
1098      catch ( Exception e )
          {
1100         System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
          }
1102   }
       }
```

Listing C.33: ServerSetup.java

```
       package serverapp;
  2
       import javax.swing.*;
  4    import javax.swing.border.*;
       import javax.swing.event.*;
  6    import java.awt.*;
       import java.util.Properties;
  8    import java.io.*;
       import java.awt.event.ActionListener;
 10    import java.awt.event.*;
       /*
 12     * The server setup dialog box.
        * Creation date: (28-05-00 20:58:57)
 14     * @author:
        */
 16    public class ServerSetup extends JDialog
       {
 18       public class OKButtonHandler implements ActionListener
          {
 20          public void actionPerformed( ActionEvent ae )
             {
 22             submitAndClose();
             }
 24       };

 26       public class CancelButtonHandler implements ActionListener
          {
 28          public void actionPerformed( ActionEvent ae )
             {
 30             cancelAndClose();
             }
 32       };

 34
          private JButton ivjCancelButton = null;
 36       private JPanel ivjJDialogContentPane = null;
          private JPanel ivjJPanel1 = null;
 38       private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
          private JButton ivjOKButton = null;
 40       private JComboBox ivjProtocolComboBox = null;
          private JLabel ivjProtocolLabel = null;
 42       private JTextField ivjServerNameField = null;
          private JLabel ivjServerNameLabel = null;
 44       private JLabel ivjServerPortLabel = null;
          private JTextField ivjServerPortField = null;
 46       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();
            ivjJPanel1.setName("JPanel1");
            ivjJPanel1.setLayout(getJPanel1FlowLayout());
            getJPanel1().add(getOKButton(), getOKButton().getName());
            getJPanel1().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 ivjJPanel1;
   }
   /**
    * Return the JPanel1FlowLayout property value.
    * @return java.awt.FlowLayout
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private java.awt.FlowLayout getJPanel1FlowLayout() {
      java.awt.FlowLayout ivjJPanel1FlowLayout = null;
      try {
         /* Create part */
         ivjJPanel1FlowLayout = new java.awt.FlowLayout();
         ivjJPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
      };
      return ivjJPanel1FlowLayout;
   }
   /**
    * 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}
            ivjOKButton.addActionListener( new OKButtonHandler() );
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjOKButton;
   }
   /**
    * Return the ProtocolComboBox property value.
    * @return javax.swing.JComboBox
    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JComboBox getProtocolComboBox() {
      if (ivjProtocolComboBox == null) {
         try {
            ivjProtocolComboBox = new javax.swing.JComboBox();
            ivjProtocolComboBox.setName("ProtocolComboBox");
            // user code begin {1}
            // user code end
         } catch (java.lang.Throwable ivjExc) {
```

```
            // user code begin {2}
246         // user code end
            handleException(ivjExc);
248      }
      }
250   return ivjProtocolComboBox;
   }
252 /**
    * Return the ProtocolLabel property value.
254  * @return javax.swing.JLabel
    */
256 /* WARNING: THIS METHOD WILL BE REGENERATED. */
   private javax.swing.JLabel getProtocolLabel() {
258   if (ivjProtocolLabel == null) {
         try {
260         ivjProtocolLabel = new javax.swing.JLabel();
            ivjProtocolLabel.setName("ProtocolLabel");
262         ivjProtocolLabel.setText("Connection:");
            // 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 ivjProtocolLabel;
272 }
   /**
274  * Return the ServerNameField property value.
    * @return javax.swing.JTextField
276  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
278 private javax.swing.JTextField getServerNameField() {
      if (ivjServerNameField == null) {
280      try {
            ivjServerNameField = new javax.swing.JTextField();
282         ivjServerNameField.setName("ServerNameField");
            // user code begin {1}
284         // user code end
         } catch (java.lang.Throwable ivjExc) {
286         // user code begin {2}
            // user code end
288         handleException(ivjExc);
         }
290   }
      return ivjServerNameField;
292 }
   /**
294  * Return the ServerNameLabel property value.
    * @return javax.swing.JLabel
296  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
298 private javax.swing.JLabel getServerNameLabel() {
      if (ivjServerNameLabel == null) {
300      try {
            ivjServerNameLabel = new javax.swing.JLabel();
302         ivjServerNameLabel.setName("ServerNameLabel");
            ivjServerNameLabel.setText("Server␣name:");
304         // user code begin {1}
            // user code end
306      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
308         // user code end
            handleException(ivjExc);
310      }
```

```
      }
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;
            constraintsServerNameField.weightx = 1.0;
            constraintsServerNameField.weighty = 0.2;
            constraintsServerNameField.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getServerNameField(), constraintsServerNameField);

            java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
            constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
            constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsServerPortLabel.weighty = 0.2;
            constraintsServerPortLabel.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getServerPortLabel(), constraintsServerPortLabel);

            java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
            constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
            constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsServerPortField.weightx = 1.0;
            constraintsServerPortField.weighty = 0.2;
            constraintsServerPortField.ipadx = 100;
            constraintsServerPortField.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getServerPortField(), constraintsServerPortField);

            java.awt.GridBagConstraints constraintsProtocolLabel = new java.awt.GridBagConstraints();
            constraintsProtocolLabel.gridx = 0; constraintsProtocolLabel.gridy = 2;
            constraintsProtocolLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsProtocolLabel.weighty = 0.2;
            constraintsProtocolLabel.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getProtocolLabel(), constraintsProtocolLabel);

            java.awt.GridBagConstraints constraintsProtocolComboBox = new java.awt.GridBagConstraints();
            constraintsProtocolComboBox.gridx = 1; constraintsProtocolComboBox.gridy = 2;
            constraintsProtocolComboBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
            constraintsProtocolComboBox.weightx = 1.0;
            constraintsProtocolComboBox.weighty = 0.2;
            constraintsProtocolComboBox.insets = new java.awt.Insets(6, 6, 6, 6);
            getTopPanel().add(getProtocolComboBox(), constraintsProtocolComboBox);
            // user code begin {1}
            Border matteBorder = BorderFactory.createMatteBorder(6,6,6,6,ivjTopPanel.getBackground());
            Border lineBorder = BorderFactory.createLineBorder(Color.gray);
            Border compoundBorder = BorderFactory.createCompoundBorder(lineBorder, matteBorder);
            Border secondCompoundBorder = BorderFactory.createCompoundBorder(matteBorder, compoundBorder);
            ivjTopPanel.setBorder(secondCompoundBorder);
            // user code end
         } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
         }
      }
      return ivjTopPanel;
   }
   /**
    * 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 {
444      // user code begin {1}
         // user code end
446      setName("ServerSetup");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
448      setSize(458, 365);
         setTitle("Server␣setup");
450      setContentPane(getJDialogContentPane());
      } catch (java.lang.Throwable ivjExc) {
452      handleException(ivjExc);
      }
454   // user code begin {2}
      setLocationRelativeTo( getOwner() );
456   // user code end
   }
458 /**
    * main entrypoint - starts the part when it is run as an application
460 * @param args java.lang.String[]
    */
462 public static void main(java.lang.String[] args) {
      try {
464      ServerSetup aServerSetup;
         aServerSetup = new ServerSetup();
466      aServerSetup.setModal(true);
         aServerSetup.addWindowListener(new java.awt.event.WindowAdapter() {
468         public void windowClosing(java.awt.event.WindowEvent e) {
               System.exit(0);
470         };
         });
472      aServerSetup.setVisible(true);
      } catch (Throwable exception) {
474      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
476   }
   }
478 /**
    * Insert the method's description here.
480 * Creation date: (05-08-00 20:33:03)
    */
482 public void show()
   {
484   getServerNameField().setText( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_NAME
         () ));
      getServerPortField().setText( ServerInfo.getPreferences().getProperty( ServerConstants.getSERVER_PORT
         () ));
486   if ( getProtocolComboBox().getItemCount() == 0 )
      {
488      for ( int i = 0; i < ServerConstants.getCONNECTION_TYPES().length; i++ )
         {
490         getProtocolComboBox().addItem( ServerConstants.getCONNECTION_TYPES()[i] );
         }
492   }
      super.show();
494 }
   /**
496 * Insert the method's description here.
    * Creation date: (05-08-00 20:54:36)
498 */
   public void submitAndClose()
500 {
      ServerInfo.getPreferences().setProperty( ServerConstants.getSERVER_NAME(), getServerNameField().
         getText() );
502   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() {
```

```
46     return name;
   }
48 /**
    * Insert the method's description here.
50  * Creation date: (05-09-00 14:24:28)
    * @return java.lang.String
52  */
   public java.lang.String getPath() {
54     return path;
   }
56 /**
    * Insert the method's description here.
58  * Creation date: (19-08-00 13:08:13)
    * @return int
60  */
   public int getUncompressedLength() {
62     return uncompressedLength;
   }
64 /**
    * Insert the method's description here.
66  * Creation date: (19-08-00 13:10:51)
    * @param newCompressedContents byte[]
68  */
   public void setCompressedContents(byte[] newCompressedContents) {
70     compressedContents = newCompressedContents;
   }
72 /**
    * Insert the method's description here.
74  * Creation date: (19-08-00 13:08:45)
    * @param newCompressedLength int
76  */
   public void setCompressedLength(int newCompressedLength) {
78     compressedLength = newCompressedLength;
   }
80 /**
    * Insert the method's description here.
82  * Creation date: (28-10-00 00:18:50)
    * @param newName java.lang.String
84  */
   public void setName(java.lang.String newName) {
86     name = newName;
   }
88 /**
    * Insert the method's description here.
90  * Creation date: (05-09-00 14:24:28)
    * @param newPath java.lang.String
92  */
   public void setPath(java.lang.String newPath) {
94     path = newPath;
   }
96 /**
    * Insert the method's description here.
98  * Creation date: (19-08-00 13:08:13)
    * @param newUncompressedLength int
100 */
   public void setUncompressedLength(int newUncompressedLength) {
102    uncompressedLength = newUncompressedLength;
   }
104 }
```

Listing C.36: ConfirmationBox.java

```
   package peerviewmisc;
 2
    import javax.swing.event.*;
 4  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);
```

```
      }
138   }
      return ivjButtonPanel;
140 }
    /**
142  * Return the ButtonPanelFlowLayout property value.
     * @return java.awt.FlowLayout
144  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
146 private java.awt.FlowLayout getButtonPanelFlowLayout() {
      java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
148   try {
        /* Create part */
150     ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
        ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
152   } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
154   };
      return ivjButtonPanelFlowLayout;
156 }
    /**
158  * Return the CancelButton property value.
     * @return javax.swing.JButton
160  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
162 private javax.swing.JButton getCancelButton() {
      if (ivjCancelButton == null) {
164     try {
          ivjCancelButton = new javax.swing.JButton();
166       ivjCancelButton.setName("CancelButton");
          ivjCancelButton.setText("Cancel");
168       // user code begin {1}
          ivjCancelButton.addActionListener( new CancelButtonHandler() );
170       // user code end
        } catch (java.lang.Throwable ivjExc) {
172       // user code begin {2}
          // user code end
174       handleException(ivjExc);
        }
176   }
      return ivjCancelButton;
178 }
    /**
180  * Return the IconLabel property value.
     * @return javax.swing.JLabel
182  */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
184 private javax.swing.JLabel getIconLabel() {
      if (ivjIconLabel == null) {
186     try {
          ivjIconLabel = new javax.swing.JLabel();
188       ivjIconLabel.setName("IconLabel");
          ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
              general/ContextualHelp24.gif")));
190       ivjIconLabel.setText("");
          // user code begin {1}
192       // user code end
        } catch (java.lang.Throwable ivjExc) {
194       // user code begin {2}
          // user code end
196       handleException(ivjExc);
        }
198   }
      return ivjIconLabel;
200 }
    /**
```

```
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.*;
```

```
 4   import java.util.Hashtable;

 6   /**
     * This class implements a container used for transmitting data between clients.
 8   * Creation date: (04-07-00 17:10:18)
     * @author:
10   */

12  public class DataPackage implements Serializable
    {
14     private final static java.lang.String AUTHOR_NAME = "Author␣name";
       private final static java.lang.String DOCPATH = "Document␣name␣and␣path";
16     private final static java.lang.String DOCNAME = "Document␣name␣(without␣path)";
       private final static java.lang.String DESCRIPTION = "Description";
18     private java.util.Properties properties = new java.util.Properties();
       private java.lang.Object contents = null;
20     private final static java.lang.String WORKGROUP_ID = "Workgroup␣ID";
       private final static java.lang.String MESSAGE_ID = "Message␣ID";
22     private final static java.lang.String CLIENT_NAME = "Client␣name";
    /**
24   * DataPackage constructor comment.
     */
26  public DataPackage() {
       super();
28  }
    /**
30   * Insert the method's description here.
     * Creation date: (27-10-00 22:45:37)
32   * @return java.lang.String
     */
34  public final static java.lang.String getAUTHOR_NAME() {
       return AUTHOR_NAME;
36  }
    /**
38   * Insert the method's description here.
     * Creation date: (29-10-00 19:05:42)
40   * @return java.lang.String
     */
42  public final static java.lang.String getCLIENT_NAME() {
       return CLIENT_NAME;
44  }
    /**
46   * Insert the method's description here.
     * Creation date: (27-10-00 22:59:23)
48   * @return java.lang.Object
     */
50  public java.lang.Object getContents() {
       return contents;
52  }
    /**
54   * Insert the method's description here.
     * Creation date: (27-10-00 22:52:15)
56   * @return java.lang.String
     */
58  public final static java.lang.String getDESCRIPTION() {
       return DESCRIPTION;
60  }
    /**
62   * Insert the method's description here.
     * Creation date: (27-10-00 22:49:36)
64   * @return java.lang.String
     */
66  public final static java.lang.String getDOCNAME() {
       return DOCNAME;
68  }
    /**
```

```
70   * Insert the method's description here.
     * Creation date: (27-10-00 22:49:02)
72   * @return java.lang.String
     */
74  public final static java.lang.String getDOCPATH() {
       return DOCPATH;
76  }
    /**
78   * Insert the method's description here.
     * Creation date: (28-10-00 00:03:49)
80   * @return java.lang.String
     */
82  public final static java.lang.String getMESSAGE_ID() {
       return MESSAGE_ID;
84  }
    /**
86   * Insert the method's description here.
     * Creation date: (27-10-00 22:55:47)
88   * @return java.util.Properties
     */
90  public java.util.Properties getProperties() {
       return properties;
92  }
    /**
94   * Insert the method's description here.
     * Creation date: (27-10-00 23:18:45)
96   * @return java.lang.String
     */
98  public final static java.lang.String getWORKGROUP_ID() {
       return WORKGROUP_ID;
100 }
    /**
102  * Insert the method's description here.
     * Creation date: (27-10-00 22:59:23)
104  * @param newContents java.lang.Object
     */
106 public void setContents(java.lang.Object newContents) {
       contents = newContents;
108 }
    /**
110  * Insert the method's description here.
     * Creation date: (27-10-00 22:55:47)
112  * @param newProperties java.util.Properties
     */
114 public void setProperties(java.util.Properties newProperties) {
       properties = newProperties;
116 }
    }
```

Listing C.38: DeleteGroup.java

```
    package peerviewmisc;
2
    /**
4    * This class implements the delete group dialog box.
     * Creation date: (29-06-00 16:19:12)
6    * @author:
     */
8   public class DeleteGroup extends DisplayGroup {
    /**
10   * DeleteGroup constructor comment.
     */
12  public DeleteGroup() {
       super();
14     initialize();
    }
16  /**
```

```
     * 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. */
344 private javax.swing.JPanel getMainPanel() {
      if (ivjMainPanel == null) {
346      try {
            ivjMainPanel = new javax.swing.JPanel();
348         ivjMainPanel.setName("MainPanel");
            ivjMainPanel.setLayout(new java.awt.GridBagLayout());
350
            java.awt.GridBagConstraints constraintsGroupNameLabel = new java.awt.GridBagConstraints();
352         constraintsGroupNameLabel.gridx = 0; constraintsGroupNameLabel.gridy = 0;
            constraintsGroupNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
354         constraintsGroupNameLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getGroupNameLabel(), constraintsGroupNameLabel);
356
            java.awt.GridBagConstraints constraintsDescriptionLabel = new java.awt.GridBagConstraints();
358         constraintsDescriptionLabel.gridx = 0; constraintsDescriptionLabel.gridy = 2;
            constraintsDescriptionLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
360         constraintsDescriptionLabel.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getDescriptionLabel(), constraintsDescriptionLabel);
362
            java.awt.GridBagConstraints constraintsGroupNameField = new java.awt.GridBagConstraints();
364         constraintsGroupNameField.gridx = 1; constraintsGroupNameField.gridy = 0;
            constraintsGroupNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
366         constraintsGroupNameField.anchor = java.awt.GridBagConstraints.EAST;
            constraintsGroupNameField.weightx = 1.0;
368         constraintsGroupNameField.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getGroupNameField(), constraintsGroupNameField);
370
            java.awt.GridBagConstraints constraintsDescriptionScrollPane = new java.awt.GridBagConstraints()
                ;
372         constraintsDescriptionScrollPane.gridx = 1; constraintsDescriptionScrollPane.gridy = 2;
            constraintsDescriptionScrollPane.fill = java.awt.GridBagConstraints.BOTH;
374         constraintsDescriptionScrollPane.weightx = 1.0;
            constraintsDescriptionScrollPane.weighty = 1.0;
376         constraintsDescriptionScrollPane.insets = new java.awt.Insets(4, 4, 4, 4);
            getMainPanel().add(getDescriptionScrollPane(), constraintsDescriptionScrollPane);
378         // user code begin {1}
            getMainPanel().setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ));
380         // user code end
         } catch (java.lang.Throwable ivjExc) {
382         // user code begin {2}
            // user code end
384         handleException(ivjExc);
         }
386   }
      return ivjMainPanel;
388 }
   /**
390  * Return the OKButton property value.
     * @return javax.swing.JButton
392  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
394 public javax.swing.JButton getOKButton() {
      if (ivjOKButton == null) {
396      try {
            ivjOKButton = new javax.swing.JButton();
398         ivjOKButton.setName("OKButton");
            ivjOKButton.setMnemonic('o');
400         ivjOKButton.setText("OK");
            // user code begin {1}
402         ivjOKButton.addActionListener( new OKButtonHandler() );
            // user code end
404      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
406         // user code end
            handleException(ivjExc);
```

```
408         }
      }
410      return ivjOKButton;
   }
412   /**
    * Insert the method's description here.
414    * Creation date: (03-07-00 20:56:36)
    * @return peerviewmisc.WorkGroup
416    */
   public WorkGroup getWorkGroup() {
418      return workGroup;
   }
420   /**
    * Called whenever the part throws an exception.
422    * @param exception java.lang.Throwable
    */
424   private void handleException(java.lang.Throwable exception) {

426      /* Uncomment the following lines to print uncaught exceptions to stdout */
      // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
428      // exception.printStackTrace(System.out);
   }
430   /**
    * Initialize the class.
432    */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
434   private void initialize() {
      try {
436         // user code begin {1}
         // user code end
438         setName("DisplayGroup");
         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
440         setSize(426, 240);
         setContentPane(getJDialogContentPane());
442      } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
444      }
      // user code begin {2}
446      setModal( true );
      // set to true as default to ensure that cancelling using the upper right corner X doesn't result in
           a spurious entry
448      setIsDiscarded( true );
      setLocationRelativeTo( getOwner() );
450
      // user code end
452   }
   /**
454    * Insert the method's description here.
    * Creation date: (26-07-00 19:10:38)
456    * @return boolean
    */
458   public boolean isIsDiscarded() {
      return isDiscarded;
460   }
   /**
462    * main entrypoint - starts the part when it is run as an application
    * @param args java.lang.String[]
464    */
   public static void main(java.lang.String[] args) {
466   /* try {
         DisplayGroup aDisplayGroup;
468         aDisplayGroup = new DisplayGroup();
         aDisplayGroup.setModal(true);
470         aDisplayGroup.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
472               System.exit(0);
```

```
        };
      });
      aDisplayGroup.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: (26-07-00 17:28:18)
 * @param newGroupDirectory clientapp.GroupDirectory
 */
public void setGroupDirectory(clientapp.GroupDirectory newGroupDirectory) {
  groupDirectory = newGroupDirectory;
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 19:10:38)
 * @param newIsDiscarded boolean
 */
public void setIsDiscarded(boolean newIsDiscarded) {
  isDiscarded = newIsDiscarded;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 20:56:36)
 * @param newWorkGroup peerviewmisc.WorkGroup
 */
public void setWorkGroup(WorkGroup newWorkGroup)
{
  workGroup = newWorkGroup;

  getGroupNameField().setText( workGroup.getGroupName() );
  getDescriptionField().setText( workGroup.getDescription() );
}
/**
 * Insert the method's description here.
 * Creation date: (27-07-00 00:02:29)
 */
public abstract void submitAndClose();
}
```

Listing C.40: DocPackage.java

```
package peerviewmisc;

/**
 * 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.
 * Creation date: (11-10-00 12:03:22)
 * @author:
 */
public class DocPackage implements java.io.Serializable {
  private java.lang.String docNameAndPath = null;
  private java.awt.Dimension size = null;
  private java.awt.Dimension position = null;
  private java.lang.String NodeLocked = null;
/**
 * DocPackage constructor comment.
 */
public DocPackage() {
  super();
}
/**
```

```
     * 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;
```

```
120         aEditGroup = new EditGroup();
            aEditGroup.setModal(true);
122         aEditGroup.addWindowListener(new java.awt.event.WindowAdapter() {
               public void windowClosing(java.awt.event.WindowEvent e) {
124               System.exit(0);
               };
126         });
            aEditGroup.setVisible(true);
128      } catch (Throwable exception) {
            System.err.println("Exception␣occurred␣in␣main()␣of␣serverapp.EditGroup");
130         exception.printStackTrace(System.out);
      }
132   }
      /**
134    * Insert the method's description here.
       * Creation date: (04-07-00 00:15:52)
136    */
      public void submitAndClose()
138   {
         String groupName = ivjGroupNameField.getText();
140      String description = ivjDescriptionField.getText();
         ErrorBox errorBox = new ErrorBox();
142
         if ( groupName.length() == 0 )
144      {
            errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
146         errorBox.show();
            return;
148      }
         getWorkGroup().setDescription( ivjDescriptionField.getText() );
150      getWorkGroup().setGroupName( ivjGroupNameField.getText() );
         setIsDiscarded( false );
152      dispose();
      }
154   }
```

Listing C.42: ErrorBox.java

```
      package peerviewmisc;
 2
      import java.applet.*;
 4    import java.awt.*;
      import java.awt.event.*;
 6    import javax.swing.*;
      import javax.swing.border.*;
 8    import javax.swing.table.*;
      import javax.swing.event.*;
10
      /**
12     * The dialog box displayed when the user is notified of an error.
       * Creation date: (07-06-00 00:37:01)
14     * @author:
       */
16    public class ErrorBox extends JDialog {

18       public class CloseButtonHandler implements ActionListener
         {
20          public void actionPerformed( ActionEvent ae )
            {
22             dispose();
            }
24       }
         private JPanel ivjJDialogContentPane = null;
26       private JPanel ivjButtonPanel = null;
         private FlowLayout ivjButtonPanelFlowLayout = null;
28       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) {
294      handleException(ivjExc);
   }
296   // user code begin {2}
   // user code end
298 }
   /**
300  * main entrypoint - starts the part when it is run as an application
   * @param args java.lang.String[]
302  */
   public static void main(java.lang.String[] args) {
304   try {
         ErrorBox aErrorBox;
306      aErrorBox = new ErrorBox();
         aErrorBox.setModal(true);
308      aErrorBox.addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
310            System.exit(0);
            };
312      });
         aErrorBox.setText( "ASFADSRFDFDSFADSFDSFADFDASFSASDFSDFASFSDFADDSFASDFSFADFADFADFDSFDSFASDFS" );
314
         aErrorBox.setVisible(true);
316
   } catch (Throwable exception) {
318      System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
         exception.printStackTrace(System.out);
320   }
   }
322 /**
    * Insert the method's description here.
324  * Creation date: (10-07-00 22:52:54)
    * @param newText java.lang.String
326  */
   public void setText(String newText)
328 {
   getMessagePanel().setText( newText );
330 getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel
        ().getFontMetrics( getMessagePanel().getFont() )));
   getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
332 pack();
   setLocationRelativeTo( getOwner() );
334 }
   }
```

Listing C.43: HelpDialog.java

```
   package peerviewmisc;
 2
   import javax.swing.tree.*;
 4
   /**
 6  * This class has now been superseded by a JavaHelp window but was originally intended as an
        implementation of the PeerView
    * help system.
 8  * Creation date: (29-05-00 12:39:46)
    * @author:
10  */
   public class HelpDialog extends javax.swing.JDialog {
12   private javax.swing.JPanel ivjHelpDialog = null;
     private javax.swing.JPanel ivjJPanel1 = null;
14   private java.awt.FlowLayout ivjJPanel1FlowLayout = null;
     private javax.swing.JSplitPane ivjJSplitPane1 = null;
16   private javax.swing.JTextPane ivjJTextPane1 = null;
     private javax.swing.JButton ivjOKButton = null;
18   IvjEventHandler ivjEventHandler = new IvjEventHandler();
```

```
class IvjEventHandler implements java.awt.event.ActionListener {
     public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == HelpDialog.this.getOKButton())
           connEtoM1(e);
     };
   };
   private javax.swing.JTree ivjTopicTree = null;
/**
 * HelpDialog constructor comment.
 */
public HelpDialog() {
   super();
   initialize();
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Dialog
 */
public HelpDialog(java.awt.Dialog owner) {
   super(owner);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public HelpDialog(java.awt.Dialog owner, String title) {
   super(owner, title);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public HelpDialog(java.awt.Dialog owner, String title, boolean modal) {
   super(owner, title, modal);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public HelpDialog(java.awt.Dialog owner, boolean modal) {
   super(owner, modal);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Frame
 */
public HelpDialog(java.awt.Frame owner) {
   super(owner);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public HelpDialog(java.awt.Frame owner, String title) {
   super(owner, title);
}
/**
 * HelpDialog constructor comment.
 * @param owner java.awt.Frame
 * @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());
            // 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 JPanel1FlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getJPanel1FlowLayout() {
    java.awt.FlowLayout ivjJPanel1FlowLayout = null;
    try {
        /* Create part */
        ivjJPanel1FlowLayout = new java.awt.FlowLayout();
        ivjJPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
    } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
    };
    return ivjJPanel1FlowLayout;
}
/**
 * Return the JSplitPane1 property value.
 * @return javax.swing.JSplitPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JSplitPane getJSplitPane1() {
    if (ivjJSplitPane1 == null) {
        try {
            ivjJSplitPane1 = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
            ivjJSplitPane1.setName("JSplitPane1");
            getJSplitPane1().add(getJTextPane1(), "bottom");
            getJSplitPane1().add(getTopicTree(), "top");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    }
    return ivjJSplitPane1;
}
/**
 * Return the JTextPane1 property value.
 * @return javax.swing.JTextPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextPane getJTextPane1() {
    if (ivjJTextPane1 == null) {
        try {
            ivjJTextPane1 = new javax.swing.JTextPane();
            ivjJTextPane1.setName("JTextPane1");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
```

```
    }
    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);
}
```

```
   /**
284  * Initialize the class.
     */
286  /* WARNING: THIS METHOD WILL BE REGENERATED. */
     private void initialize() {
288     try {
           // user code begin {1}
290        // user code end
           setName("HelpDialog");
292        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
           setSize(426, 240);
294        setTitle("Help");
           setContentPane(getHelpDialog());
296        initConnections();
        } catch (java.lang.Throwable ivjExc) {
298        handleException(ivjExc);
        }
300     // user code begin {2}
        setLocationRelativeTo( getOwner() );
302     initTree();
        // user code end
304  }
     /**
306  * Insert the method's description here.
     * Creation date: (02-09-00 21:20:16)
308  */
     public void initTree()
310  {
        javax.swing.tree.DefaultMutableTreeNode rootNode = new javax.swing.tree.DefaultMutableTreeNode( "Not␣
            yet␣fully␣implemented" );
312     ( (DefaultTreeModel) getTopicTree().getModel() ).setRoot( rootNode );
     }
314  /**
     * main entrypoint - starts the part when it is run as an application
316  * @param args java.lang.String[]
     */
318  public static void main(java.lang.String[] args) {
        try {
320        HelpDialog aHelpDialog;
           aHelpDialog = new HelpDialog();
322        aHelpDialog.setModal(true);
           aHelpDialog.addWindowListener(new java.awt.event.WindowAdapter() {
324           public void windowClosing(java.awt.event.WindowEvent e) {
                 System.exit(0);
326           };
           });
328        aHelpDialog.setVisible(true);
        } catch (Throwable exception) {
330        System.err.println("Exception␣occurred␣in␣main()␣of␣javax.swing.JDialog");
           exception.printStackTrace(System.out);
332     }
     }
334  }
```

Listing C.44: Message.java

```
     package peerviewmisc;
 2
      import java.io.Serializable;
 4    import java.util.Vector;

 6    /**
      * This class implements a representation of the messages that can be displayed in the message area.
 8    * Creation date: (10-07-00 00:07:52)
      * @author:
10    */
     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;
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 01:42:07)
 * @return java.lang.String
 */
public java.lang.String getDocument() {
   return document;
}
/**
 * Insert the method's description here.
 * Creation date: (19-07-00 18:14:33)
 * @return java.lang.String[]
 */
public final static java.lang.String[] getFIELD_NAMES() {
   return FIELD_NAMES;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:11:49)
 * @return java.lang.String
 */
public java.lang.String getMessageID()
{  // 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
   // identically entitled messages at the exact same millisecond ?) and aesthetically unpleasing when
        presented
   // to the user.
   return this.toString() + getDocument();
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:11:02)
 * @return java.lang.String
 */
public java.lang.String getTitle() {
   return title;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:09:03)
 * @param newAuthor java.lang.String
 */
public void setAuthor(java.lang.String newAuthor) {
   author = newAuthor;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 00:08:45)
 * @param newContents java.lang.String
 */
public void setContents(java.lang.String newContents) {
   contents = newContents;
}
/**
 * Insert the method's description here.
 * Creation date: (16-07-00 10:40:40)
 * @param newCreationDate java.util.GregorianCalendar
 */
public void setCreationDate(java.util.Date newCreationDate) {
   creationDate = newCreationDate;
}
/**
 * Insert the method's description here.
```

```
140   * Creation date: (19-07-00 17:29:31)
      * @param newCreator java.lang.String
142   */
      public void setCreator(java.lang.String newCreator) {
144      creator = newCreator;
      }
146   /**
      * Insert the method's description here.
148   * Creation date: (10-07-00 01:42:07)
      * @param newDocument java.lang.String
150   */
      public void setDocument(java.lang.String newDocument) {
152      document = newDocument;
      }
154   /**
      * Insert the method's description here.
156   * Creation date: (10-07-00 00:11:02)
      * @param newTitle java.lang.String
158   */
      public void setTitle(java.lang.String newTitle) {
160      title = newTitle;
      }
162   /**
      * Insert the method's description here.
164   * Creation date: (11-07-00 12:53:01)
      * @return java.lang.String
166   */
      public String toString()
168   {
         return title + ",␣by␣" + author + "␣on␣" + creationDate.toString();
170   }
      }
```

Listing C.45: MessageBox.java

```
      package peerviewmisc;
 2
      import java.awt.event.*;
 4
      /**
 6    * This class implements a box which is used for displaying brief messages.
      * Creation date: (25-07-00 13:50:50)
 8    * @author:
      */
10    public class MessageBox extends javax.swing.JDialog
      {
12       public class CloseButtonHandler implements ActionListener
         {
14          public void actionPerformed( ActionEvent ae )
            {
16             dispose();
            }
18       }

20       private javax.swing.JPanel ivjButtonPanel = null;
         private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
22       private javax.swing.JButton ivjCloseButton = null;
         private javax.swing.JLabel ivjIconLabel = null;
24       private javax.swing.JPanel ivjJDialogContentPane = null;
         private javax.swing.JPanel ivjJDialogContentPane1 = null;
26       private javax.swing.JPanel ivjMainPanel = null;
         private javax.swing.JTextPane ivjMessagePanel = null;
28    /**
      * MessageBox constructor comment.
30    */
      public MessageBox() {
32       super();
```

```
    initialize();
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 */
public MessageBox(java.awt.Dialog owner) {
    super(owner);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 */
public MessageBox(java.awt.Dialog owner, String title) {
    super(owner, title);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 * @param title java.lang.String
 * @param modal boolean
 */
public MessageBox(java.awt.Dialog owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Dialog
 * @param modal boolean
 */
public MessageBox(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 */
public MessageBox(java.awt.Frame owner) {
    super(owner);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 */
public MessageBox(java.awt.Frame owner, String title) {
    super(owner, title);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public MessageBox(java.awt.Frame owner, String title, boolean modal) {
    super(owner, title, modal);
}
/**
 * MessageBox constructor comment.
 * @param owner java.awt.Frame
 * @param modal boolean
 */
public MessageBox(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}
```

```
   /**
100  * Return the ButtonPanel property value.
     * @return javax.swing.JPanel
102  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
104 private javax.swing.JPanel getButtonPanel() {
      if (ivjButtonPanel == null) {
106      try {
            ivjButtonPanel = new javax.swing.JPanel();
108         ivjButtonPanel.setName("ButtonPanel");
            ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
110         getButtonPanel().add(getCloseButton(), getCloseButton().getName());
            // user code begin {1}
112         // user code end
         } catch (java.lang.Throwable ivjExc) {
114         // user code begin {2}
            // user code end
116         handleException(ivjExc);
         }
118   }
      return ivjButtonPanel;
120 }
   /**
122  * Return the ButtonPanelFlowLayout property value.
     * @return java.awt.FlowLayout
124  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
126 private java.awt.FlowLayout getButtonPanelFlowLayout() {
      java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
128   try {
         /* Create part */
130      ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
132   } catch (java.lang.Throwable ivjExc) {
         handleException(ivjExc);
134   };
      return ivjButtonPanelFlowLayout;
136 }
   /**
138  * Return the CloseButton property value.
     * @return javax.swing.JButton
140  */
   /* WARNING: THIS METHOD WILL BE REGENERATED. */
142 private javax.swing.JButton getCloseButton() {
      if (ivjCloseButton == null) {
144      try {
            ivjCloseButton = new javax.swing.JButton();
146         ivjCloseButton.setName("CloseButton");
            ivjCloseButton.setMnemonic('C');
148         ivjCloseButton.setText("Close");
            // user code begin {1}
150         ivjCloseButton.addActionListener( new CloseButtonHandler() );
            // user code end
152      } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
154         // user code end
            handleException(ivjExc);
156      }
      }
158   return ivjCloseButton;
   }
160 /**
     * Return the IconLabel property value.
162  * @return javax.swing.JLabel
     */
164 /* 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) {
```

```
74      super(owner, title, modal);
    }
76  /**
     * NewGroup constructor comment.
78   * @param owner java.awt.Frame
     * @param modal boolean
80   */
    public NewGroup(java.awt.Frame owner, boolean modal) {
82      super(owner, modal);
    }
84  /**
     * Called whenever the part throws an exception.
86   * @param exception java.lang.Throwable
     */
88  private void handleException(java.lang.Throwable exception) {

90      /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
92      // exception.printStackTrace(System.out);
    }
94  /**
     * Initialize the class.
96   */
    /* WARNING: THIS METHOD WILL BE REGENERATED. */
98  private void initialize() {
        try {
100         // user code begin {1}
            // user code end
102         setName("NewGroup");
            setModal(true);
104     } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
106     }
        // user code begin {2}
108     setTitle( "New group" );
        editable( true );
110     getOKButton().setText( "Submit" );
        getOKButton().setMnemonic( 's' );
112     getCancelButton().setText( "Discard" );
        getCancelButton().setMnemonic( 'd' );
114
        // user code end
116 }
    /**
118  * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
120  */
    public static void main(java.lang.String[] args) {
122     try {
            NewGroup aNewGroup;
124         aNewGroup = new NewGroup();
            aNewGroup.setModal(true);
126         aNewGroup.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
128                 System.exit(0);
                };
130         });
            aNewGroup.setVisible(true);
132     } catch (Throwable exception) {
            System.err.println("Exception occurred in main() of serverapp.NewGroup");
134         exception.printStackTrace(System.out);
        }
136 }
    /**
138  * Insert the method's description here.
     * Creation date: (04-07-00 00:15:52)
```

```
140   */
      public void submitAndClose()
142   {
         String groupName = ivjGroupNameField.getText();
144      String description = ivjDescriptionField.getText();
         ErrorBox errorBox = new ErrorBox();
146
         if ( groupName.length() == 0 )
148      {
            errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
150         errorBox.show();
            return;
152      }
         if ( getGroupDirectory().groupExists( groupName, description )== true )
154      {
            errorBox.setText( SharedConstants.getGROUP_EXISTS_MESSAGE() );
156         errorBox.show();
            return;
158      }
         getWorkGroup().setDescription( ivjDescriptionField.getText() );
160      getWorkGroup().setGroupName( ivjGroupNameField.getText() );
         setIsDiscarded( false );
162      dispose();
      }
164   }
```

Listing C.47: Pair.java

```
      package peerviewmisc;
 2
      /**
 4     * A container primitive mapping a key to a value.
       * Creation date: (20-09-00 02:31:55)
 6     * @author:
       */
 8    public class Pair implements java.io.Serializable {
         private java.lang.String key = null;
10       private java.lang.String value = null;
      /**
12     * Pair constructor comment.
       */
14    public Pair() {
         super();
16    }
      /**
18     * Insert the method's description here.
       * Creation date: (20-09-00 02:36:16)
20     * @param keyArg java.lang.String
       * @param valueArg java.lang.String
22     */
      public Pair(String keyArg, String valueArg)
24    {
         key = keyArg;
26       value = valueArg;
      }
28    /**
       * Insert the method's description here.
30     * Creation date: (20-09-00 02:32:13)
       * @return java.lang.String
32     */
      public java.lang.String getKey() {
34       return key;
      }
36    /**
       * Insert the method's description here.
38     * Creation date: (20-09-00 02:34:30)
       * @return java.lang.String
```

```
40   */
     public java.lang.String getValue() {
42     return value;
     }
44   /**
      * Insert the method's description here.
46    * Creation date: (20-09-00 02:32:13)
      * @param newKey java.lang.String
48    */
     public void setKey(java.lang.String newKey) {
50     key = newKey;
     }
52   /**
      * Insert the method's description here.
54    * Creation date: (20-09-00 02:34:30)
      * @param newValue java.lang.String
56    */
     public void setValue(java.lang.String newValue) {
58     value = newValue;
     }
60   }
```

Listing C.48: SharedConstants.java

```
     package peerviewmisc;
 2
      import javax.swing.border.*;
 4    import javax.swing.BorderFactory.*;
      import com.sun.media.jsdt.*;
 6    import java.util.Hashtable;

 8    /**
      * An entity object containing constants shared by the client and server applications.
10    * Creation date: (30-06-00 21:39:29)
      * @author:
12    */
     public class SharedConstants {
14     protected final static int BORDER_WIDTH = 6;
       protected final static java.lang.String GROUP_MANAGEMENT_SESSION_NAME = "group␣management";
16     protected final static java.lang.String DEFAULT_SERVER_NAME = "localhost";
       protected final static int DEFAULT_SERVER_PORT = 4461;
18     protected final static java.lang.String CONNECTION_TYPE = "Connection␣type";
       protected final static java.lang.String REQUEST_GROUP_DIRECTORY = "request␣group␣directory";
20
       protected final static java.lang.String SERVER_NAME = "Server␣name";
22     protected final static java.lang.String SERVER_PORT = "Server␣port";
       protected final static java.lang.String DEFAULT_CONNECTION_TYPE = "socket";
24     private final static java.lang.String REQUEST_ADD_GROUP = "Request␣add␣group";
       private final static java.lang.String GROUP_DIRECTORY = "Group␣directory";
26     private final static java.lang.String NEW_DOCUMENT = "New␣document";
       private final static java.lang.String GROUP_CHANNEL_NAME = "GroupChannel";
28     private final static java.lang.String WORKGROUP_CHANNEL_NAME = "Workgroup␣channel";
       private final static java.lang.String CLIENT_ADDED_DOCUMENTS = "Client␣added␣documents";
30     private final static java.lang.String REQUEST_TOPIC_TREE = "Request␣topic␣tree";
       private final static java.lang.String TOPIC_TREE = "Topic␣tree";
32     private final static java.lang.String REQUEST_MESSAGE = "Request␣message";
       private final static java.lang.String MESSAGE = "Message";
34     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";
36     private final static java.lang.String TOPIC_TREE_UPDATE = "Topic␣tree␣update";
       private final static java.lang.String DELETE_MESSAGE = "Delete␣message";
38     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";
40     protected final static java.lang.String DEFAULT_CLIENT_NAME = "defaultName";
       private final static java.lang.String WORKGROUP_UPDATE_FROM_CLIENT = "Workgroup␣update␣from␣client";
```

```
42      private final static java.lang.String WORKGROUP_UPDATE = "Workgroup␣update";
        private final static java.lang.String CLIENT_LEFT_GROUP = "Client␣left␣group";
44      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.";
46      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.";
48      private final static java.lang.String REQUEST_REMOVE_GROUP = "Remove␣workgroup␣from␣directory";

50        private final static java.lang.String SOCKET = "socket";
        private final static java.lang.String HTTP = "http";
52      private final static java.lang.String[] CONNECTION_TYPES = {getSOCKET(),getHTTP()};

54      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" },
56        { 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"},
58        { 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" },
60        { 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" },
62        { 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" },
64        { 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" },
66        { 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" },
68        { 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" },
70        { new TimedOutException().getClass().toString(), "An␣operation␣timed␣out␣before␣it␣could␣be␣
              completed" },
        };
72      private final static java.util.Hashtable JSDT_EXCEPTION_MESSAGES_TABLE = new Hashtable();
        private final static int NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS = 50;
74      private final static java.lang.String COMPRESSION_LEVEL = "Compression␣level";
        private final static int DEFAULT_COMPRESSION_LEVEL = java.util.zip.Deflater.BEST_COMPRESSION;
76      private final static java.lang.String DISCUSSIONS_FILE_NAME = "discuss.dat";
        private final static java.lang.String SOCKET_ARGUMENT = "-socket";
78      private final static java.lang.String HTTP_ARGUMENT = "-http";

80      private final static java.lang.String REGISTRY_PORT = "Registry␣port";
        private final static java.lang.String DEFAULT_REGISTRY_PORT = "4561";
82      private final static java.lang.String WORKGROUP_INCONSISTENT = "Workgroup␣inconsistent";
        private final static java.lang.String CLIENT_JOINED_GROUP = "Client␣joined␣group";
84      private final static java.lang.String NEW_DOCUMENTS = "New␣documents";
        private final static java.lang.String REMOVE_DOCUMENTS = "Remove␣documents";
86      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.";
88    /**
       * SharedConstants constructor comment.
90     */
```

```
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;
}
/**
 * Insert the method's description here.
 * Creation date: (10-08-00 06:56:22)
 * @return int
 */
public final static int getNOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS() {
    return NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS;
}
/**
 * Insert the method's description here.
 * Creation date: (12-09-00 10:16:20)
 * @return java.lang.String
 */
public final static java.lang.String getREGISTRY_PORT() {
    return REGISTRY_PORT;
}
/**
 * Insert the method's description here.
 * Creation date: (26-07-00 02:41:39)
 * @return java.lang.String
 */
public final static java.lang.String getREMOVE_DOCUMENT() {
    return REMOVE_DOCUMENT;
}
/**
 * Insert the method's description here.
 * Creation date: (19-09-00 19:15:50)
 * @return java.lang.String
 */
public final static java.lang.String getREMOVE_DOCUMENTS() {
    return REMOVE_DOCUMENTS;
}
/**
 * Insert the method's description here.
 * Creation date: (04-07-00 17:33:01)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_ADD_GROUP() {
    return REQUEST_ADD_GROUP;
}
/**
 * Insert the method's description here.
 * Creation date: (03-07-00 12:39:46)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_GROUP_DIRECTORY() {
    return REQUEST_GROUP_DIRECTORY;
}
/**
 * Insert the method's description here.
 * Creation date: (10-07-00 02:16:46)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_MESSAGE() {
    return REQUEST_MESSAGE;
}
/**
 * Insert the method's description here.
 * Creation date: (27-07-00 00:26:45)
 * @return java.lang.String
 */
public final static java.lang.String getREQUEST_REMOVE_GROUP() {
    return REQUEST_REMOVE_GROUP;
}
```

```
    /**
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;
    }
```

```
172  /**
      * Create and return a unique identifier for the workgroup
174   * Creation date: (17-07-00 17:53:57)
      * @return java.lang.String
176   */
     public String getWorkGroupID()
178  {
        return getGroupName() + getDescription() + getCreator();
180  }
     /**
182   * Insert the method's description here.
      * Creation date: (16-07-00 11:01:38)
184   * @param newCreationDate java.util.Date
      */
186  public void setCreationDate(java.util.Date newCreationDate) {
        creationDate = newCreationDate;
188  }
     /**
190   * Insert the method's description here.
      * Creation date: (17-07-00 17:12:13)
192   * @param newCreator java.lang.String
      */
194  public void setCreator(java.lang.String newCreator) {
        creator = newCreator;
196  }
     /**
198   * Insert the method's description here.
      * Creation date: (05-07-00 21:32:10)
200   * @param newDescription java.lang.String
      */
202  public void setDescription(java.lang.String newDescription) {
        description = newDescription;
204  }
     /**
206   * Insert the method's description here.
      * Creation date: (19-09-00 17:40:00)
208   * @param newDocuments java.util.Hashtable
      */
210  public void setDocuments(java.util.Hashtable newDocuments) {
        documents = newDocuments;
212  }
     /**
214   * Insert the method's description here.
      * Creation date: (05-07-00 21:25:46)
216   * @param newGroupName java.lang.String
      */
218  public void setGroupName(java.lang.String newGroupName) {
        groupName = newGroupName;
220  }
     /**
222   * Insert the method's description here.
      * Creation date: (19-09-00 18:45:20)
224   * @param newNumberOfDocuments int
      */
226  public void setNumberOfDocuments(int newNumberOfDocuments) {
        numberOfDocuments = newNumberOfDocuments;
228  }
     /**
230   * Insert the method's description here.
      * Creation date: (19-09-00 17:53:36)
232   * @param newParticipants java.util.HashSet
      */
234  public void setParticipants(java.util.HashSet newParticipants) {
        participants = newParticipants;
236  }
     /**
```

```
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

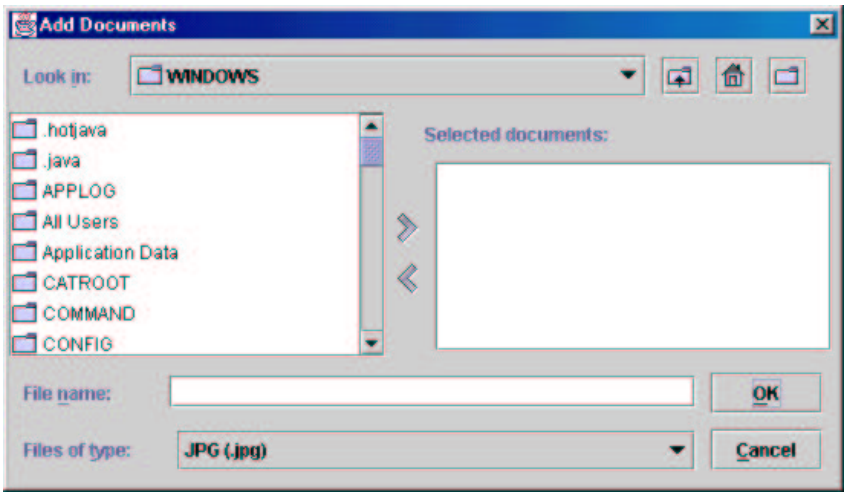
Welcome to the PeerView online help system.

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.

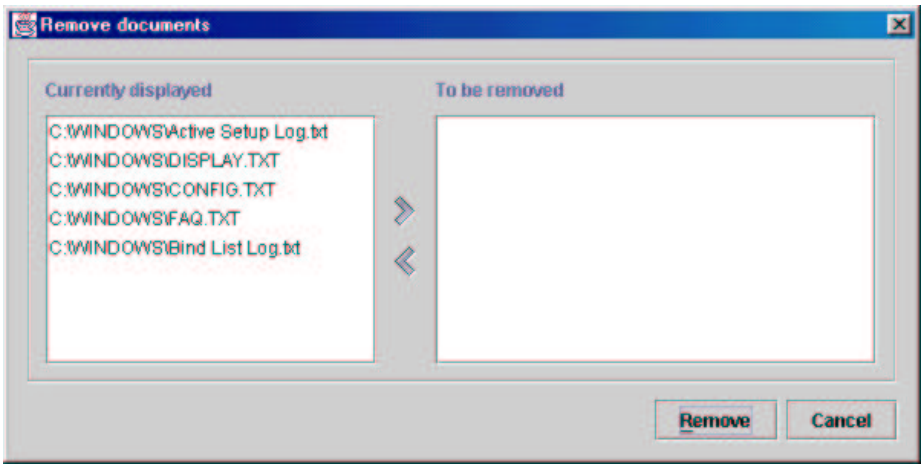
Click OK to add the files in the "Selected documents" list box to the panorama.

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.

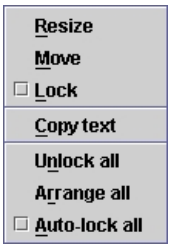
2) Double click the left mouse button

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.

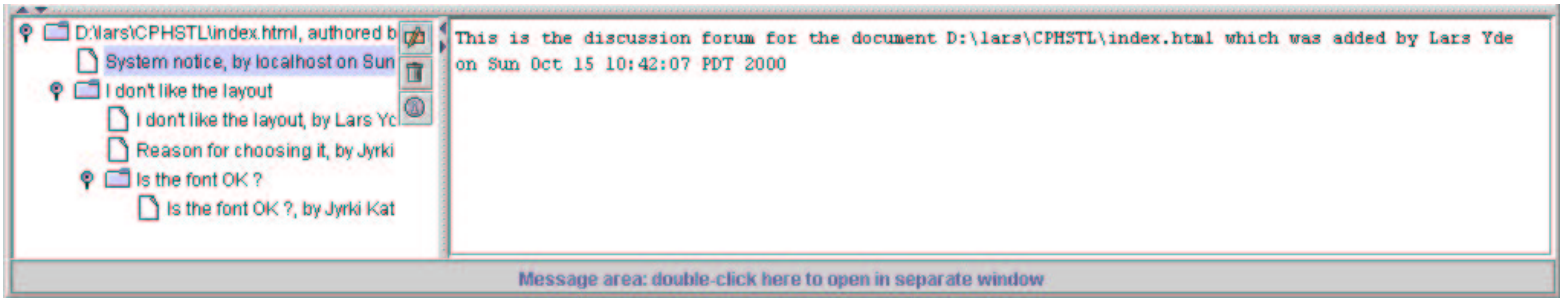
Auto−lock all

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

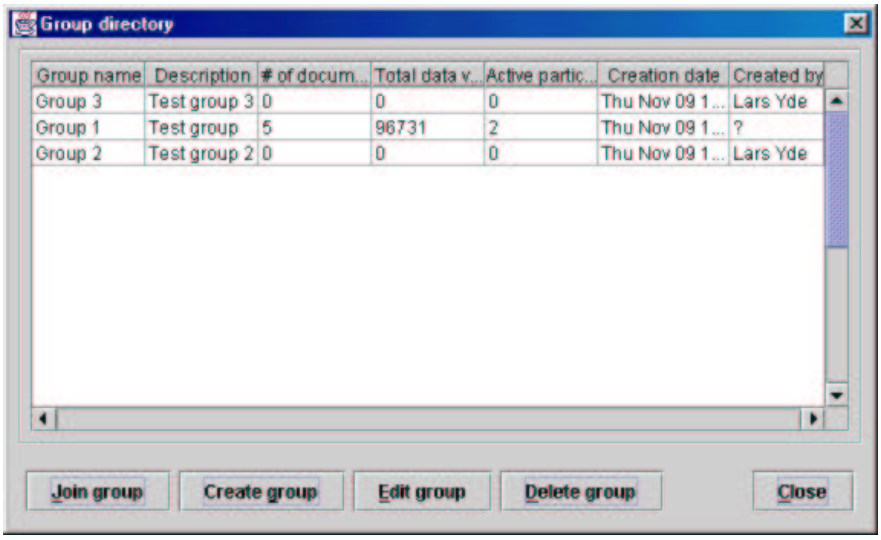
To adjust the size of the discussion forum itself, use the divider bar which separates it from the panorama.

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.

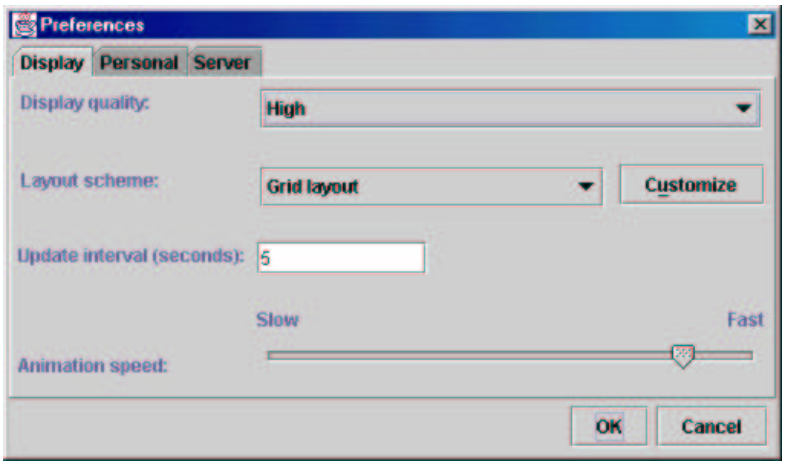
Close the group directory box.

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.

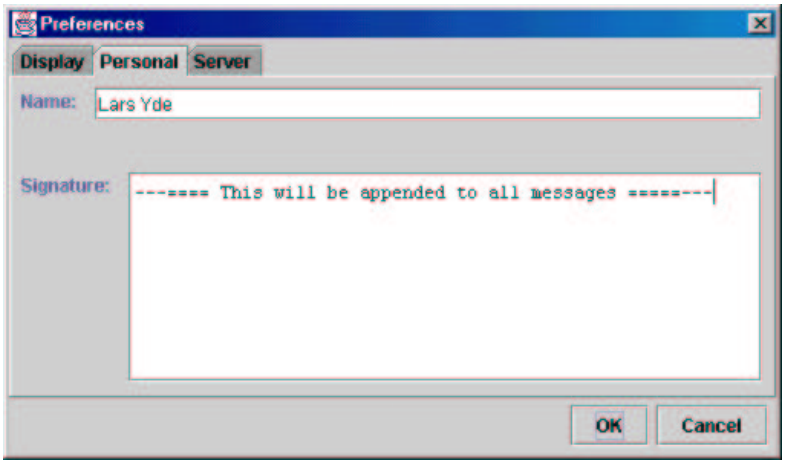
Update interval

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:

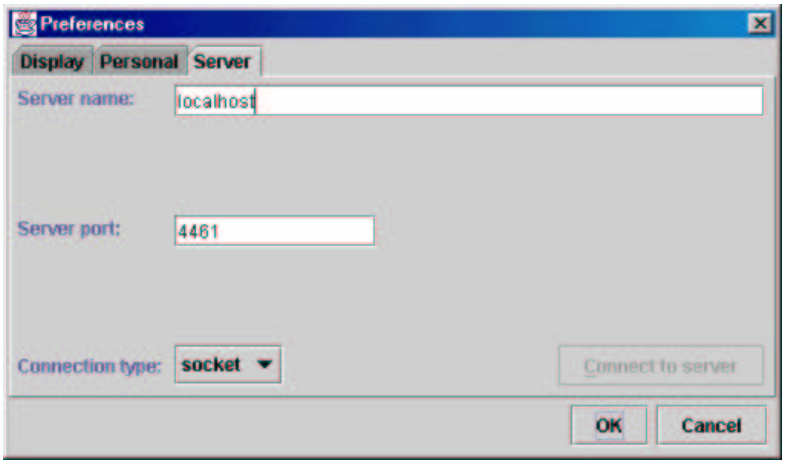
Name

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.