

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 be 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

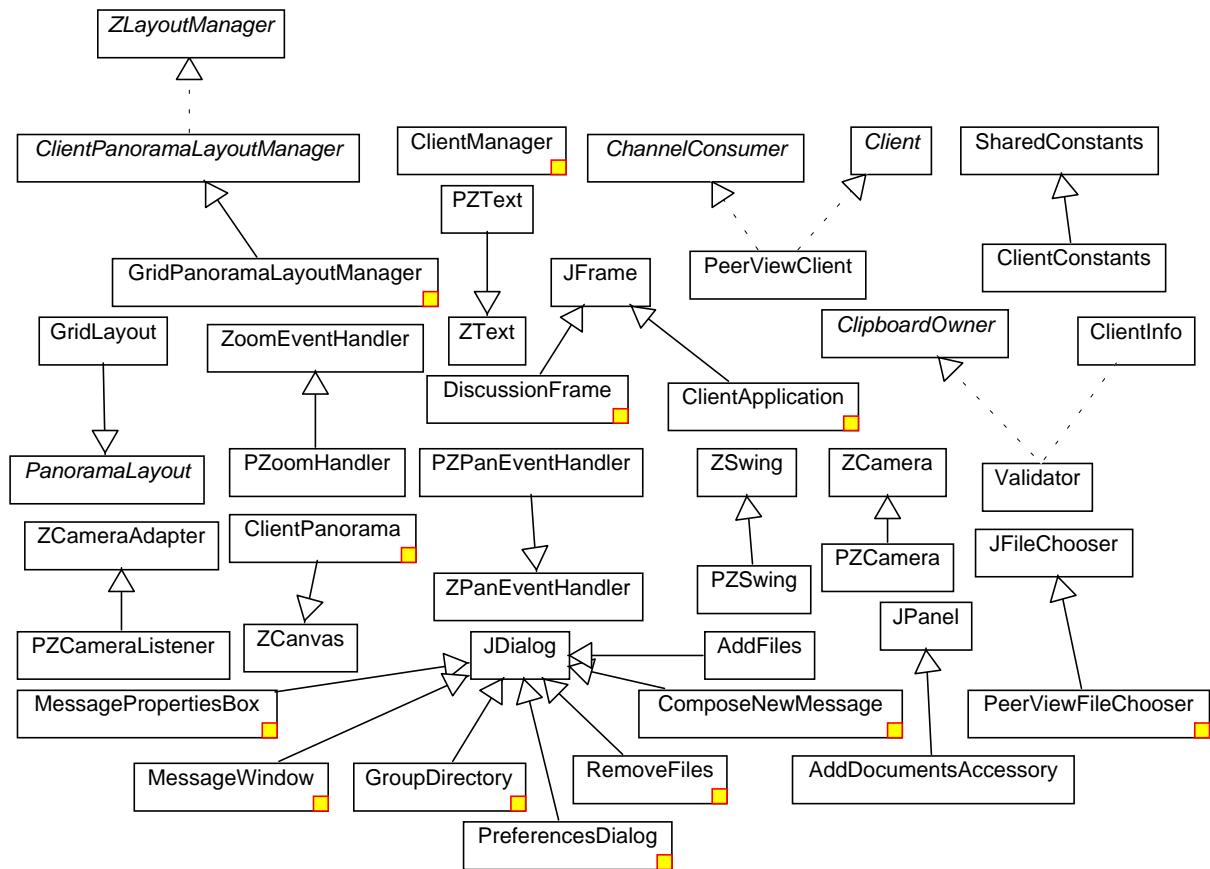


Figure A.1: Class diagram for the PeerView client application classes.

A.4 Class diagram for package serverapp

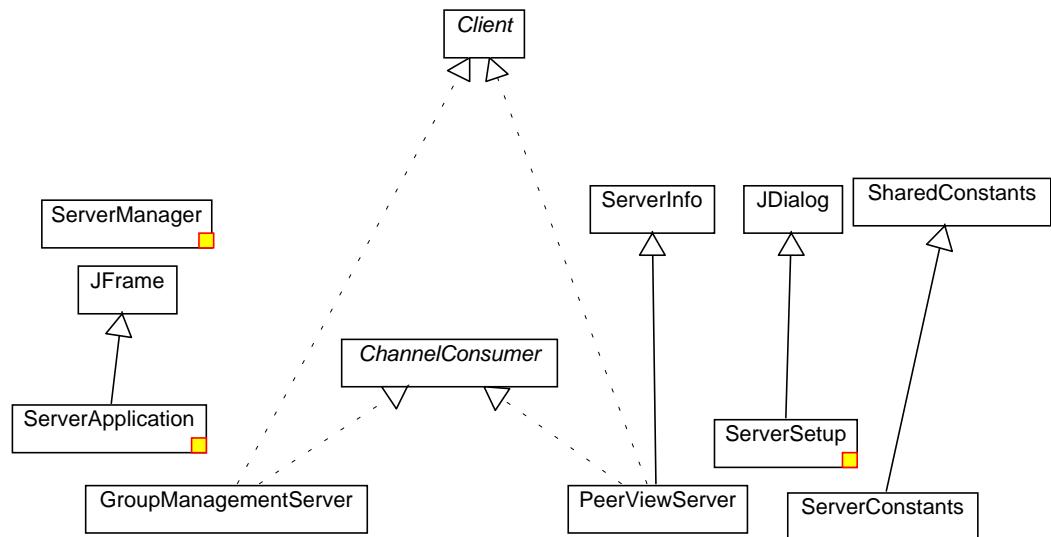


Figure A.2: Class diagram for the PeerView server application classes.

A.5 Class diagram for package peerviewmisc

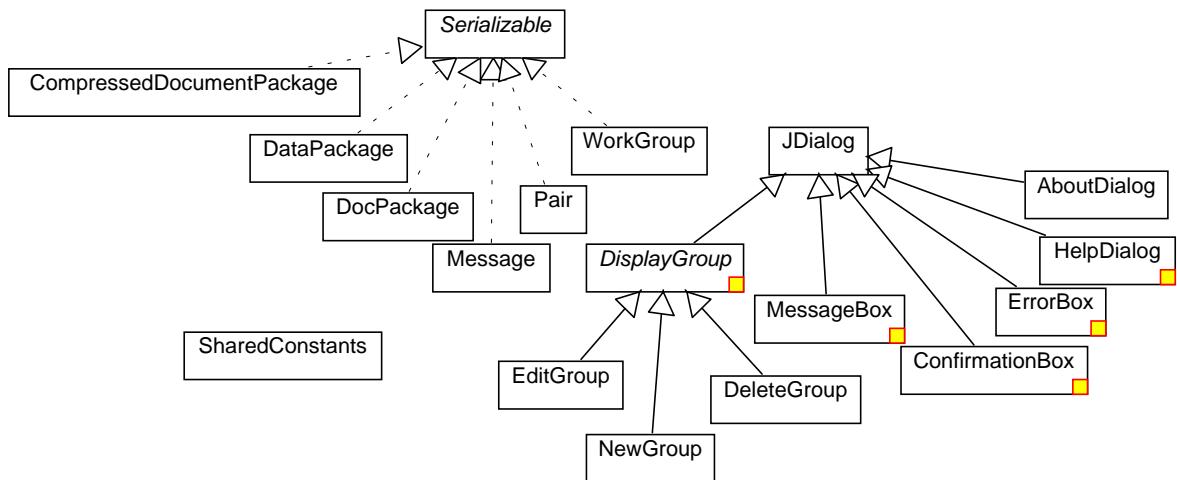


Figure A.3: Class diagram for shared and miscellaneous classes.

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 transcript 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 arbejdsmiljø.

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 vælge “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.

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 arbejdsmiljø.

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, markér 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.

15. Dette trin er valgfrit – du behøver ikke udføre det. Markér en portion tekst i dokumentet som du kunne tænke dig at give en kommentar til, højre-klik musen og vælg “Copy text” fra den menu der dukker op (pop-up menu).
16. Markér et element i træet i hovedvinduets nederste venstre hjørne. Klik på den knap der lader dig skrive et indlæg (“Compose new message”). Brug hjælpen fra “Help” menuen hvis du har problemer.
17. Skriv en kommentar til dokumentet og indsæt gerne evt. tekst valgt i trin 15 ved at højreklikke i det område hvor du skriver teksten og vælge “Paste text”.
18. Afslut boksen ved at vælge “Submit”.
19. Dobbeltklik området udenfor dokumenterne i hovedvinduets midte således at der zoomes ud til et oversigtsniveau hvor du kan overskue alle dokumenterne.
20. Højreklik ovenpå et tilfældigt dokument og vælg “Move”. Flyt dokumentet til en ny position. Brug hjælpesystemet hvis du har brug for det. Højreklik igen på dokumentet og sæt et kryds i feltet “Lock” hvis det ikke allerede er gjort.
21. Højreklik på et andet dokument og vælg “Resize”. Giv dokumentet en ny størrelse.
22. Højreklik på et tilfældigt dokument og vælg “Arrange all”.
23. Vælg dokumentet “test_gruppe.txt” fra “Documents in panorama” boksen. Hvis der er sket en ændring siden du sidst valgte det, så gå videre. Ellers vent til der er sket en ændring.
24. Åbn boksen “Remove documents” og fjern alle dine dokumenter fra systemet. Brug hjælpesystemet hvis du har brug for det.
25. Afslut PeerView ved at vælge “Exit” fra menuen.

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 assessment 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 assessment 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 if 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.2 Transsscript

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 afdvikles.
9. Lars: I skal bare downloade med
10. Rikke: Jeg skal bare trykke her [peger på skærmen]
11. Lars: Ja, tryk bare ...
12. Thomas: Programmet kører direkte [InstallAnywhere installationsprogrammet]
13. Lars: Ja, det kører direkte [over netværksforbindelsen], ja
14. Thomas: Den spørger hvor man skal placere det [installationsprogrammet] ...køre det eller gemme det på...?
15. Lars: øhm, ja du kan bare køre det der ...det tager så lidt tid.
[ca. 10 minutters pause]
16. Lars: Proceduren her var så ikke så svær at gennemføre. Det der var det største problem var sidt jeg kunne se at indtaste adressen [url'en].
17. Rikke: Det er fordi tastaturet har en anden [opsætning (end den hun kender fra sit arbejde)] ...
18. Lars: [forklarer hvorfor InstallAnywhere er brugt til at lave installationsprogrammet]
19. Paul: Jeg vidste ikke helt hvad for en man skulle vælge af de der [installationsmuligheder] ...det var kun fordi thomas havde sagt det [at der skulle downloades med Java Virtual Machine].
20. Lars: Ja, i virkeligheden skulle folk så helst finde ud af det af sig selv. Jeg står selvfølgelig ikke og kan fortælle dem det når de skal downloade, så det skal helst gøres klarere. Client, det er ikke særlig ...det kræver en teknisk indsigt at se at det er den klient der skal køres. Så det er faktisk en dårlig betegnelse.

21. Lars: Ja, det her Java ... Jeg kunne have valgt at downloade det hele på forhånd, men det lod jeg være med for jeg syntes også i skulle igennem den her del af det.
22. Thomas: Skal jeg taste det der R-y-g-e-r-4 ind ... det er jo tastet ind.
23. Lars: Jamen hvis det er det så er det fordi den husker indstillingerne fra dengang jeg testede [prøveinstallerede] det.
24. Thomas: da jeg trykkede OK, så gik den ud af programmet ... Jeg flyttede nogle filer over [d.v.s mellem listboksene i "Add Documents" boksen], ikke
25. Lars: Ja,
26. Thomas: da jeg så trykkede OK gik den ud af programmet.
27. Lars: BEMÆRKNING. Man må ikke tilføje kataloger i "Add Files" bokse.
28. Rikke: Hvad skal man ...
29. Lars: Du skal først vælge dokumenter heroppe fra ... ja det har du gjort.
30. Lars: Jamen du har ikke forbindelse til serveren endnu. Vi gør lige det vi lukker det ned igen og starter det op. Fordi den har ikke kunnet ... den har mistet forbindelsen. Det sker nogen gange.
31. Rikke: Jeg kan bare starte på [trin] 14 igen.
32. Lars: Ja, den skulle gerne gå igang med at tilføje ting igen nu.
33. Lars: Det er fordi [servernavnet] skal være skrevet med stort [begyndelsesbogstav] ... jeg tror det er derfor ... ja.
34. Lars: Prøv at gøre det samme forløb som du gjorde sidst her. Det var simpelthen fordi den [servernavnet] var med småt der, tror jeg.
35. Thomas: Jeg overførte nogle filer fra [available documents] boksen [til selected documents boksen].
36. Lars: Der var ingen kataloger imellem ?
37. Thomas: Nej
38. Lars: Det var derfor så [at programmet terminerede tidligere] ... så skal du bare køre videre.
39. Thomas: Nå , okay.
40. Rikke: Kan 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 oplosninger 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 gör 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 også på et tidspunkt skulle det blive sådan at den giver en liste over dem der har deltaget, men det gör 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 så 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 nøprede 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 du 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 saadiges egnet til.

På den måde vil der blive skabt en redhård for nye bruger, som måske ikke har den store indsigt i, hvad det hell går ud på.

Dog skal det siges, at jeg godt er klar, at programmet konvender 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 hjælpestystemet 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	*	*				Langsamt

Kommentarer

PeerView er en prototype og tanken er at det på et senere tidspunkt skal videreføres 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 HARDE UNDREVEJS SKYLDNES TIL ØELS, AT "TINGENE" STOD PÅ ENGELSKE (BETYKONINGEN "FØRSVANDT" FØR MIG).
- JEG SKULLE BRUGE MUSEN MEGET - GENVEJSTASTERNE "FØRSVANDT"

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/temner 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

X Brugte den ikke

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 installere prg. i Windows og brug principippet til at udvæksle problemløsninger etc.
- Jeg har indtryk af at man hurtig vil blive "DUS" med programmet, og vil glad for det i det rette miljø.
- Jeg syntes også at de engelske ord er ret tekniske for en "normal" bruger.

Thomas

B.3.2 English version of PeerView questionnaire for first participant

Usability questions

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

	Agree entirely	Agree	Neutral	Disagree 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 be used in before it is put to use. That way a novice user who may not have great insight into the purpose of it all will have a guideline. I do realize, though, that the programme is aimed at a particular type of user who will be well aware of what the programme is for.

B.3.3 English version of PeerView questionnaire for second participant

Usability questions

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

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

Evaluation questions

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

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

Comments

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

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

B.3.4 English version of PeerView questionnaire for third participant

Usability questions

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

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

Evaluation questions

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

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

Comments

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

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

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

Appendix C

Source code

C.1 Introduction

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

C.2 List of listings

Listings

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

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

Listing C.1: AddDocumentsAccessory.java

```

1 package clientapp;
2
3 /**
4  * This class defines the right hand side interface components of the
5  * "Add documents" dialog box, i.e. the "Selected documents" list box
6  * and the two arrow buttons used for transferring files into and out
7  * from it.
8  * Creation date: (15-06-00 15:29:40)
9  * @author:
10 */
11
12 import javax.swing.*;
13 import java.awt.*;
14 import java.io.*;
15 import java.awt.event.*;
16
17 public class AddDocumentsAccessory extends JPanel {
18     private JButton ivjAddButton = null;
19     private JPanel ivjButtonPanel = null;
20     private JList ivjDocumentList = null;
21     private JScrollPane ivjScrollPane = null;
22     private JButton ivjRemoveButton = null;
23     private javax.swing.DefaultListModel selectedDocuments;
24     private JLabel ivjJLabel = null;

```

```

26 /**
27  * AddDocumentsAccessory constructor comment.
28 */
29 public AddDocumentsAccessory() {
30     super();
31     initialize();
32 }
33 /**
34  * AddDocumentsAccessory constructor comment.
35  * @param layout java.awt.LayoutManager
36 */
37 public AddDocumentsAccessory(java.awt.LayoutManager layout) {
38     super(layout);
39 }
40 /**
41  * AddDocumentsAccessory constructor comment.
42  * @param layout java.awt.LayoutManager
43  * @param isDoubleBuffered boolean
44 */
45 public AddDocumentsAccessory(java.awt.LayoutManager layout, boolean isDoubleBuffered) {
46     super(layout, isDoubleBuffered);
47 }
48 /**
49  * AddDocumentsAccessory constructor comment.
50  * @param isDoubleBuffered boolean
51 */
52 public AddDocumentsAccessory(boolean isDoubleBuffered) {
53     super(isDoubleBuffered);
54 }
55 /**
56  * Return the AddButton property value.
57  * @return javax.swing.JButton
58 */
59 /* WARNING: THIS METHOD WILL BE REGENERATED. */
60 public javax.swing.JButton getAddButton() {
61     if (ivjAddButton == null) {
62         try {
63             ivjAddButton = new javax.swing.JButton();
64             ivjAddButton.setName("AddButton");
65             ivjAddButton.setToolTipText("Add_to_list");
66             ivjAddButton.setMnemonic('a');
67             ivjAddButton.setText("");
68             ivjAddButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
69                 navigation/Forward24.gif")));
70             ivjAddButton.setBorderPainted(false);
71             ivjAddButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
72             // user code begin {1}
73             // user code end
74         } catch (java.lang.Throwable ivjExc) {
75             // user code begin {2}
76             // user code end
77             handleException(ivjExc);
78         }
79     }
80     return ivjAddButton;
81 }
82 /**
83  * Return the ButtonPanel property value.
84  * @return javax.swing.JPanel
85 */
86 /* WARNING: THIS METHOD WILL BE REGENERATED. */
87 private javax.swing.JPanel getButtonPanel() {
88     if (ivjButtonPanel == null) {
89         try {
90             ivjButtonPanel = new javax.swing.JPanel();
91             ivjButtonPanel.setName("ButtonPanel");

```

```

        ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
92    ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
93    ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
94
95    java.awt.GridBagConstraints constraintsAddButton = new java.awt.GridBagConstraints();
96    constraintsAddButton.gridx = 0; constraintsAddButton.gridy = 0;
97    getButtonPanel().add(getAddButton(), constraintsAddButton);
98
99    java.awt.GridBagConstraints constraintsRemoveButton = new java.awt.GridBagConstraints();
100   constraintsRemoveButton.gridx = 0; constraintsRemoveButton.gridy = 1;
101   getButtonPanel().add(getRemoveButton(), constraintsRemoveButton);
102
103   // user code begin {1}
104   // user code end
105
106 } catch (java.lang.Throwable ivjExc) {
107     // user code begin {2}
108     // user code end
109     handleException(ivjExc);
110 }
111
112 /**
113  * Return the DocumentList property value.
114  * @return javax.swing.JList
115  */
116 /* WARNING: THIS METHOD WILL BE REGENERATED. */
117 public javax.swing.JList getDocumentList() {
118
119     if (ivjDocumentList == null) {
120         try {
121             ivjDocumentList = new javax.swing.JList();
122             ivjDocumentList.setName("DocumentList");
123             ivjDocumentList.setToolTipText("Documents displayed in panorama");
124             ivjDocumentList.setBounds(0, 0, 112, 253);
125
126             // user code begin {1}
127             selectedDocuments = new DefaultListModel();
128             ivjDocumentList.setModel(selectedDocuments);
129             // user code end
130
131         } catch (java.lang.Throwable ivjExc) {
132             // user code begin {2}
133             // user code end
134             handleException(ivjExc);
135         }
136     }
137
138     return ivjDocumentList;
139 }
140 /**
141  * Return the JLabel property value.
142  * @return javax.swing.JLabel
143  */
144 /* WARNING: THIS METHOD WILL BE REGENERATED. */
145 private javax.swing.JLabel getJLabel() {
146
147     if (ivjJLabel == null) {
148         try {
149             ivjJLabel = new javax.swing.JLabel();
150             ivjJLabel.setName("JLabel");
151             ivjJLabel.setText("Selected documents:");
152
153             // user code begin {1}
154             // user code end
155             } catch (java.lang.Throwable ivjExc) {
156                 // user code begin {2}
157                 // user code end
158                 handleException(ivjExc);
159             }
160
161     }
162
163     return ivjJLabel;
164 }

```

```

  /**
   * Return the JScrollPane property value.
   * @return javax.swing.JScrollPane
   */
  /* WARNING: THIS METHOD WILL BE REGENERATED. */
  private javax.swing.JScrollPane getJScrollPane() {
    if (ivjJScrollPane == null) {
      164   try {
        165     ivjJScrollPane = new javax.swing.JScrollPane();
        166     ivjJScrollPane.setName("JScrollPane");
        167     getJScrollPane().setViewportView(getDocumentList());
        168     // user code begin {1}
        // user code end
      169     } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
      170     // user code end
        handleException(ivjExc);
      171   }
    }
    172   return ivjJScrollPane;
  }
  /**
   * Return the RemoveButton property value.
   * @return javax.swing.JButton
   */
  /* WARNING: THIS METHOD WILL BE REGENERATED. */
  public javax.swing.JButton getRemoveButton() {
    if (ivjRemoveButton == null) {
      184     try {
        185       ivjRemoveButton = new javax.swing.JButton();
        186       ivjRemoveButton.setName("RemoveButton");
        187       ivjRemoveButton.setToolTipText("Remove from list");
        188       ivjRemoveButton.setMnemonic('r');
        189       ivjRemoveButton.setText("");
        190       ivjRemoveButton.setMaximumSize(new java.awt.Dimension(29, 35));
        191       ivjRemoveButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/navigation/Back24.gif")));
        192       ivjRemoveButton.setBorderPainted(false);
        193       ivjRemoveButton.setPreferredSize(new java.awt.Dimension(29, 35));
        194       ivjRemoveButton.setMinimumSize(new java.awt.Dimension(29, 35));
        195       ivjRemoveButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
        // user code begin {1}
      196       // user code end
      197     } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
      198     }
    }
    199   return ivjRemoveButton;
  }
  /**
   * Insert the method's description here.
   * Creation date: (15-06-00 16:07:02)
   * @return javax.swing.DefaultListModel
   */
  public javax.swing.DefaultListModel getSelectedDocuments() {
    212   return selectedDocuments;
  }
  /**
   * Called whenever the part throws an exception.
   * @param exception java.lang.Throwable
   */
  216   private void handleException(java.lang.Throwable exception) {
    217     /* Uncomment the following lines to print uncaught exceptions to stdout */

```

```

222     // System.out.println("----- UNCAUGHT EXCEPTION -----");
223     // exception.printStackTrace(System.out);
224 }
225 /**
226 * Initialize the class.
227 */
228 /* WARNING: THIS METHOD WILL BE REGENERATED. */
229 private void initialize() {
230     try {
231         // user code begin {1}
232         // user code end
233         setName("AddDocumentsAccessory");
234         setLayout(new java.awt.GridBagLayout());
235         setSize(329, 319);
236
237         java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
238         constraintsButtonPanel.gridx = 0; constraintsButtonPanel.gridy = 1;
239         constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
240         constraintsButtonPanel.anchor = java.awt.GridBagConstraints.WEST;
241         constraintsButtonPanel.weightx = 1.0;
242         constraintsButtonPanel.weighty = 1.0;
243         add(getButtonPanel(), constraintsButtonPanel);
244
245         java.awt.GridBagConstraints constraintsJScrollPane = new java.awt.GridBagConstraints();
246         constraintsJScrollPane.gridx = 0; constraintsJScrollPane.gridy = 1;
247         constraintsJScrollPane.fill = java.awt.GridBagConstraints.BOTH;
248         constraintsJScrollPane.anchor = java.awt.GridBagConstraints.EAST;
249         constraintsJScrollPane.weightx = 1.0;
250         constraintsJScrollPane.weighty = 1.0;
251         constraintsJScrollPane.insets = new java.awt.Insets(4, 33, 4, 4);
252         add(getJScrollPane(), constraintsJScrollPane);
253
254         java.awt.GridBagConstraints constraintsJLabel = new java.awt.GridBagConstraints();
255         constraintsJLabel.gridx = 0; constraintsJLabel.gridy = 0;
256         constraintsJLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
257         constraintsJLabel.anchor = java.awt.GridBagConstraints.NORTH;
258         constraintsJLabel.insets = new java.awt.Insets(6, 20, 6, 6);
259         add(getJLabel(), constraintsJLabel);
260     } catch (java.lang.Throwable ivjExc) {
261         handleException(ivjExc);
262     }
263     // user code begin {2}
264     // user code end
265 }
266 /**
267 * main entrypoint - starts the part when it is run as an application
268 * @param args java.lang.String[]
269 */
270 public static void main(java.lang.String[] args) {
271     try {
272         javax.swing.JFrame frame = new javax.swing.JFrame();
273         AddDocumentsAccessory aAddDocumentsAccessory;
274         aAddDocumentsAccessory = new AddDocumentsAccessory();
275         frame.setContentPane(aAddDocumentsAccessory);
276         frame.setSize(aAddDocumentsAccessory.getSize());
277         frame.addWindowListener(new java.awt.event.WindowAdapter() {
278             public void windowClosing(java.awt.event.WindowEvent e) {
279                 System.exit(0);
280             };
281         });
282         frame.setVisible(true);
283     } catch (Throwable exception) {
284         System.err.println("Exception occurred in main() of javax.swing.JPanel");
285         exception.printStackTrace(System.out);
286     }
287 }

```

```

288 /**
 * Set the selectedDocuments property
289 * Creation date: (15-06-00 16:07:02)
 * @param newSelectedDocuments javax.swing.DefaultListModel
290 */
291 public void setSelectedDocuments(javax.swing.DefaultListModel newSelectedDocuments) {
292     selectedDocuments = newSelectedDocuments;
293 }
294 }
```

Listing C.2: AddFiles.java

```

package clientapp;
2
3 /**
4  * The document used for adding files/documents to the panorama.
5  * Creation date: (23-05-00 08:55:51)
6  * @author:
7  */
8 import javax.swing.*;
9 import java.awt.*;
10 import java.io.*;
11 import java.awt.event.*;
12 import javax.swing.filechooser.FileFilter;
13 import java.io.File.*;
14
15 public class AddFiles extends JDialog {
16
17     private JPanel ivjJDialogContentPane = null;
18     private PeerViewFileChooser ivjFileChooser = null;
19     private ClientApplication clientApplication;
20
21 /**
22  * Constructor
23  */
24 /* WARNING: THIS METHOD WILL BE REGENERATED. */
25 public AddFiles() {
26     super();
27     initialize();
28 }
29 /**
30  * AddFiles constructor comment.
31  * @param owner java.awt.Dialog
32 */
33 public AddFiles(java.awt.Dialog owner) {
34     super(owner);
35     initialize();
36 }
37 /**
38  * AddFiles constructor comment.
39  * @param owner java.awt.Dialog
40  * @param title java.lang.String
41 */
42 public AddFiles(java.awt.Dialog owner, String title) {
43     super(owner, title);
44 }
45 /**
46  * AddFiles constructor comment.
47  * @param owner java.awt.Dialog
48  * @param title java.lang.String
49  * @param modal boolean
50 */
51 public AddFiles(java.awt.Dialog owner, String title, boolean modal) {
52     super(owner, title, modal);
53 }
54 /**
55  * AddFiles constructor comment.
56  * @param owner java.awt.Dialog
57 }
```

```

56  * @param modal boolean
57  */
58 public AddFiles(java.awt.Dialog owner, boolean modal) {
59     super(owner, modal);
60 }
61 /**
62  * AddFiles constructor comment.
63  * @param owner java.awt.Frame
64  */
65 public AddFiles(java.awt.Frame owner) {
66     super(owner);
67     initialize();
68 }
69 /**
70  * AddFiles constructor comment.
71  * @param owner java.awt.Frame
72  * @param title java.lang.String
73  */
74 public AddFiles(java.awt.Frame owner, String title) {
75     super(owner, title);
76 }
77 /**
78  * AddFiles constructor comment.
79  * @param owner java.awt.Frame
80  * @param title java.lang.String
81  * @param modal boolean
82  */
83 public AddFiles(java.awt.Frame owner, String title, boolean modal) {
84     super(owner, title, modal);
85 }
86 /**
87  * AddFiles constructor comment.
88  * @param owner java.awt.Frame
89  * @param modal boolean
90  */
91 public AddFiles(java.awt.Frame owner, boolean modal) {
92     super(owner, modal);
93 }
94 /**
95  * Override the JDialog dispose method to allow the current file filter selection to be saved before
96  * disposing of the box.
97  * Creation date: (25-02-01 20:23:39)
98 */
99 public void dispose()
100 {
101     getFileChooser().setPreviousFileFilterSelection( getFileChooser().getFileFilter() );
102     super.dispose();
103 }
104 /**
105  * Insert the method's description here.
106  * Creation date: (16-06-00 08:34:47)
107  * @return clientapp.ClientApplication
108 */
109 public ClientApplication getClientApplication() {
110     return clientApplication;
111 }
112 /**
113  * Return the FileChooser property value.
114  * @return clientapp.PeerViewFileChooser
115 */
116 /* WARNING: THIS METHOD WILL BE REGENERATED. */
117 private PeerViewFileChooser getFileChooser() {
118     if (ivjFileChooser == null) {
119         try {
120             ivjFileChooser = new clientapp.PeerViewFileChooser();
121             ivjFileChooser.setName("FileChooser");

```

```

122     ivjFileChooser.setMaximumSize(new java.awt.Dimension(400, 150));
123     java.io.File ivjLocal41selectedFiles [] = {};
124     ivjFileChooser.setSelectedFiles(ivjLocal41selectedFiles);
125     ivjFileChooser.setPreferredSize(new java.awt.Dimension(400, 150));
126     ivjFileChooser.setMultiSelectionEnabled(true);
127     ivjFileChooser.setMinimumSize(new java.awt.Dimension(300, 150));
128     // user code begin {1}
129
130     // user code end
131 } catch (java.lang.Throwable ivjExc) {
132     // user code begin {2}
133     // user code end
134     handleException(ivjExc);
135 }
136 return ivjFileChooser;
}
138 /**
 * Return the JDDialogContentPane property value.
139 * @return javax.swing.JPanel
 */
140 /* WARNING: THIS METHOD WILL BE REGENERATED. */
141 private javax.swing.JPanel getJDDialogContentPane() {
142     if (ivjJDDialogContentPane == null) {
143         try {
144             ivjJDDialogContentPane = new javax.swing.JPanel();
145             ivjJDDialogContentPane.setName("JDDialogContentPane");
146             ivjJDDialogContentPane.setPreferredSize(new java.awt.Dimension(550, 350));
147             ivjJDDialogContentPane.setLayout(new java.awt.BorderLayout());
148             ivjJDDialogContentPane.setMinimumSize(new java.awt.Dimension(400, 250));
149             getFileChooser().add(ivjJDDialogContentPane, "Center");
150             // user code begin {1}
151             // user code end
152         } catch (java.lang.Throwable ivjExc) {
153             // user code begin {2}
154             // user code end
155             handleException(ivjExc);
156         }
157     }
158     return ivjJDDialogContentPane;
}
159 }
160 /**
 * Called whenever the part throws an exception.
161 * @param exception java.lang.Throwable
 */
162 private void handleException(java.lang.Throwable exception) {
163
164     /* Uncomment the following lines to print uncaught exceptions to stdout */
165     // System.out.println("----- UNCAUGHT EXCEPTION -----");
166     // exception.printStackTrace(System.out);
167 }
168 /**
 * Initialize the class.
169 */
170 /* WARNING: THIS METHOD WILL BE REGENERATED. */
171 public void initialize() {
172     try {
173         // user code begin {1}
174         // user code end
175         setName("AddFiles");
176         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
177         setResizable(true);
178         setSize(550, 318);
179         setModal(true);
180         setTitle("Add Documents");
181         setContentPane(getJDDialogContentPane());
182
183     }
184
185 }

```

```

188     } catch (java.lang.Throwable ivjExc) {
189         handleException(ivjExc);
190     }
191     // user code begin {2}
192     ivjFileChooser.setAddFilesDialog(this);
193     // default positioning puts the dialog at center of screen. The method used is a slight hack, but
194     // works fine.
195     setLocationRelativeTo( getOwner() );
196     // user code end
197 }
198 /**
199 * main entrypoint - starts the part when it is run as an application
200 * @param args java.lang.String[]
201 */
202 public static void main(java.lang.String[] args) {
203     try {
204         AddFiles aAddFiles;
205         aAddFiles = new AddFiles((JFrame)null);
206         aAddFiles.setModal(true);
207         aAddFiles.addWindowListener(new java.awt.event.WindowAdapter() {
208             public void windowClosing(java.awt.event.WindowEvent e) {
209                 System.exit(0);
210             }
211         });
212         aAddFiles.setVisible(true);
213     } catch (Throwable exception) {
214         System.err.println("Exception occurred in main() of javax.swing.JDialog");
215         exception.printStackTrace(System.out);
216     }
217 /**
218 * Insert the method's description here.
219 * Creation date: (16-06-00 08:34:47)
220 * @param newClientApplication clientapp.ClientApplication
221 */
222     public void setClientApplication(ClientApplication newClientApplication) {
223         clientApplication = newClientApplication;
224     }
225 /**
226 * Overrides the JDialog show method to ensure that init is called before the box is displayed.
227 * Creation date: (03-08-00 01:22:58)
228 */
229     public void show()
230     {
231         getFileChooser().init();
232         super.show();
233     }
234 /**
235 * Add the files selected by the user to the panorama and dispose of the dialog box.
236 * Creation date: (12-06-00 07:47:56)
237 */
238     public void submitAndClose()
239     {
240         Object selectedObjects[] = getFileChooser().getFileChooserAccessory().getSelectedDocuments().toArray
241             ();
242         File selectedFiles[] = new File[selectedObjects.length];
243         for (int i = 0; i < selectedFiles.length; i++)
244         { selectedFiles[i] = (File)selectedObjects[i]; }
245         clientApplication.getClientManager().addDocuments(selectedFiles);
246         clientApplication.repaint();
247         dispose();
248     }
249 }

```

Listing C.3: ClientApplication.java

```

1 package clientapp;
2
3 /**
4  * The main class of the PeerView client application.
5  * Creation date: (14-06-00 15:36:49)
6  * @author:
7  */
8 import java.applet.*;
9 import java.awt.*;
10 import java.awt.event.*;
11 import java.io.*;
12 import java.util.*;
13 import java.util.Properties;
14
15 import javax.swing.*;
16 import javax.swing.border.*;
17 import javax.swing.table.*;
18 import javax.swing.event.*;
19 import javax.swing.JProgressBar.*;
20 import javax.swing.Timer.*;
21
22 import edu.umd.cs.jazz.*;
23 import edu.umd.cs.jazz.util.*;
24 import edu.umd.cs.jazz.event.*;
25 import edu.umd.cs.jazz.component.*;
26 import javax.swing.tree.*;
27
28 import com.sun.media.jsdt.*;
29 import peerviewmisc.*;
30 import javax.help.*;

31 /** The client application main class */
32 public class ClientApplication extends JFrame
33 {
34     /** Listener class for activating JavaHelp system
35      */
36     public class HelpActionListener implements ActionListener
37     {
38         HelpBroker helpBroker;
39
40         public HelpActionListener( HelpBroker hb )
41         {
42             helpBroker = hb;
43         }
44
45         /** Event handler. Makes the online help window visible and centers it on screen */
46         public void actionPerformed( ActionEvent ae )
47         {
48             helpBroker.setDisplayed( true );
49             // calculate placement of help window
50             Point winLocation = getLocationOnScreen();
51             Dimension winSize = getSize();
52             int helpX = (int) ( winLocation.getX() + ( winSize.getWidth() - helpBroker.getSize().getWidth()
53                 ) / 2 );
54             int helpY = (int) ( winLocation.getY() + ( winSize.getHeight() - helpBroker.getSize().getHeight()
55                 ) / 2 );
56             helpBroker.setLocation( new Point( helpX, helpY ) );
57         }
58     };
59
60     /** Objects of this class can be used as caption item in the visible documents box in the client
61      * application toolbar */
62     public static class CaptionItem
63     {
64         String caption;
65         public String getCaption()
66         {
67
68

```

```

64         return caption;
65     }
66     public CaptionItem( String captionArg )
67     {
68         caption = captionArg;
69     }
70     public String toString()
71     {
72         return caption;
73     }
74 };
75     /** Objects of this class can be used as author items in the visible documents box in the client
76      application toolbar */
77     public static class AuthorItem
78     {
79         String authorName;
80         public String getAuthorName()
81         {
82             return authorName;
83         }
84         public AuthorItem( String authorNameArg )
85         {
86             authorName = authorNameArg;
87         }
88         public String toString()
89         {
90             return getAuthorName() + ":";  

91         }
92     };
93     /** Objects of this class can be used as document entries in the visible documents box in the client
94      application toolbar */
95     public static class DocumentItem
96     {
97         String documentID;
98         String documentName;
99
100        public String getDocumentID()
101        {
102            return documentID;
103        }
104        public String getDocumentName()
105        {
106            return documentName;
107        }
108        public DocumentItem( String documentIDArg, String documentNameArg )
109        {
110            documentID = documentIDArg;
111            documentName = documentNameArg;
112        }
113        public String toString()
114        {
115            return "\t" + getDocumentName();
116        }
117    };
118    /** Event handler for the visible documents box which is the dropdown box on the client application
119     toolbar where the
120     * names of the documents displayed in the panorama appear
121     */
122    public class VisibleDocumentsBoxItemListener implements ItemListener
123    {
124        int previousSelectedIndex = 0;
125
126        public void itemStateChanged( ItemEvent ie )
127        {
128            if ( ( isItemListenerActive() == true ) &&

```

```

126         ( ie.getStateChange() == ItemEvent.SELECTED ) || ( getDocumentBox().getSelectedIndex()
127             == 0 ))
128     {
129         getDocumentBox().hidePopup();
130         if ( ie.getItem() instanceof DocumentItem )
131         {
132             getPanoramaPanel().centerDocument( ( (DocumentItem) ie.getItem() ).getDocumentID() );
133         }
134         else if ( ie.getItem() instanceof AuthorItem )
135         {
136             int currentSelectedIndex = getDocumentBox().getSelectedIndex();
137             // if stepping through the list from above, skip to the succeeding item
138             if ( (currentSelectedIndex == previousSelectedIndex + 1) )
139             {
140                 getDocumentBox().setSelectedIndex( currentSelectedIndex + 1 );
141             }
142             // if stepping through the list from below, skip to the preceeding item
143             else if ( currentSelectedIndex == previousSelectedIndex - 1 )
144             {
145                 getDocumentBox().setSelectedIndex( currentSelectedIndex - 1 );
146             }
147             // if moving from non-adjacent item, negate selection
148             else
149             {
150                 getDocumentBox().setSelectedIndex( previousSelectedIndex );
151             }
152         }
153     }
154 };
155 /**
156  * Custom renderer for the visible documents box. The box is filled with items of classes
157  * {@link #CaptionItem CaptionItem}, {@link #AuthorItem AuthorItem}, {@link #DocumentItem DocumentItem}
158  * which require
159  * custom rendering to display with different levels of indentation and different icons.
160  */
161 public class VisibleDocumentsBoxCellRenderer implements ListCellRenderer
162 {
163     /**
164      ** Render method for the visible documents list. Renders caption, author names and document names
165      * with proper indentation and icons
166      */
167     public Component getListCellRendererComponent( JList list, Object value, int index, boolean
168         isSelected, boolean cellHasFocus )
169     {
170         JLabel renderLabel = new JLabel();
171         renderLabel.setIconTextGap( ClientConstants.getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() );
172         renderLabel.setBorder( BorderFactory.createMatteBorder( 3, 3, 3, 3, renderLabel.getBackground()
173             ) );
174         renderLabel.setMaximumSize( ClientConstants.getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() );
175         renderLabel.setPreferredSize( renderLabel.getMaximumSize() );
176         if ( value instanceof AuthorItem )
177         {
178             renderLabel.setText( ( (AuthorItem) value ).getAuthorName() + ":" );
179             renderLabel.setIcon( new ImageIcon( getClass().getResource( "/toolbarButtonGraphics/
180                 development/Host16.gif" ) ) );
181         }
182         else if ( value instanceof DocumentItem )
183         {
184             DocumentItem docItem = (DocumentItem) value;
185             String docName = docItem.getDocumentName();
186             if ( isSelected )
187             {

```

```

        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    }
193    // caption item last since it is the least likely (being unique)
194    else if ( value instanceof CaptionItem )
    {
195        renderLabel.setText( ( (CaptionItem) value ).getCaption() );
196        // renderLabel.setIcon( new javax.swing.ImageIcon( getClass().getResource( "/" +
            toolbarButtonGraphics/navigation/Home16.gif" ) ) );
197        renderLabel.setHorizontalTextPosition( SwingConstants.CENTER );
198        // renderLabel.setBorder( BorderFactory.createMatteBorder( 6, 6, 6, 6, renderLabel.getBackground
            () ) );
200    }
201    return renderLabel;
202}
204
/** Event handler for the message area at the bottom of the client application. */
206 public class MessageAreaMouseListener extends MouseAdapter
{
207
208     public void mouseClicked( MouseEvent me )
209     {
210         {
211             if ( me.getClickCount() == 2 )
212             {
213                 showMessageWindow();
214             }
215         }
216     }
217 }
218
/** Event handler for the client application main window */
220 public class ClientWindowListener extends WindowAdapter
{
221
222     public void windowActivated( WindowEvent we )
223     {
224         bringChildWindowsOnTop();
225     }
226     public void windowClosing( WindowEvent we )
227     {
228         //System.out.println(" windowClosing ");
229         try
230         {
231             terminateAndClose();
232             dispose();
233         }
234         catch ( Throwable t )
235         {
236         }
237     }
238 }
239
/** Event handler for the topic tree area of the client application main windows */
240 public class TopicTreeSelectionListener implements TreeSelectionListener
{
241
242     public void valueChanged( TreeSelectionEvent tse )
243     {
244         handleTopicTreeClick( tse.getPath() );
245     }
246 }

```

```

248  /** This is a small multi-purpose action listener which was implemented as an experiment to test
249   * whether running
250   * small tasks in the event-handling thread was feasible. It worked but interfered with the GUI.
251   */
252  public class Task implements ActionListener
253  {
254      private int currentValue;
255
256      public Task(int startValue)
257      {
258          currentValue = startValue;
259      }
260
261      public void actionPerformed(ActionEvent actionEvent)
262      {
263          updateMessageArea(currentValue);
264          // System.out.println("ACTION!");
265      }
266
267      public void updateValue(int newValue)
268      {
269          currentValue = newValue;
270      }
271
272      public void advance()
273      {
274          currentValue++;
275      }
276
277  /** This class is used to represent clients that participate in JSDT sessions. It therefore
278   * implements the JSDT Client interface.
279  */
280  public class DefaultClient implements Client
281  {
282      String name;
283
284      public DefaultClient (String nameArg )
285      {
286          name = nameArg;
287      }
288      public Object authenticate ( AuthenticationInfo ai )
289      {
290          return null;
291      }
292      public String getName ()
293      {
294          return name;
295      }
296
297      public class TopicTreeMouseAdapter extends MouseAdapter
298      {
299          public void mouseClicked( MouseEvent me )
300          {
301              // handleTopicTreeClick( me );
302          }
303      }
304
305  /** The renderer class for the discussion tree in the client application main window. Leaf nodes in
306   * the tree are
307   * rendered with the string representation of the user object embedded in the node
308  */
309  public class DiscussionTreeCellRenderer extends DefaultTreeCellRenderer
310  {

```

```

    public Component getTreeCellRendererComponent( JTree jTree, Object value, boolean selected,
                                                 boolean expanded, boolean leaf, int row, boolean hasFocus )
312    {
313        Component returnValue = super.getTreeCellRendererComponent( jTree, value, selected, expanded,
314                                                               leaf, row, hasFocus );
315        if ( leaf )
316        {
317            if ( value instanceof Message)
318                setText( value.toString() );
319            return this;
320        }
321    }
322
323    /** The event handler associated with the compose new message button in the client application
324     * message area */
325    public class ComposeNewMessageButtonHandler implements ActionListener
326    {
327        public void actionPerformed( ActionEvent ae )
328        {
329            composeNewMessage();
330        }
331    }
332
333    /** The event handler associated with the delete message button in the client application message
334     * area */
335    public class DeleteButtonHandler implements ActionListener
336    {
337        public void actionPerformed ( ActionEvent ae )
338        {
339            deleteMessage();
340        }
341    }
342
343    /** The event handler associated with the message properties button in the client application message
344     * area */
345    public class MessagePropertiesButtonHandler implements ActionListener
346    {
347        public void actionPerformed( ActionEvent ae )
348        {
349            showMessageProperties();
350        }
351    }
352
353    private JMenuItem ivjAbout = null;
354    private AboutDialog ivjAboutDialog1 = null;
355    private JMenuItem ivjAddDocuments = null;
356    private AddFiles ivjAddFiles1 = null;
357    private JMenuBar ivjClientApplicationJMenuBar = null;
358    IvjEventHandler ivjEventHandler = new IvjEventHandler();
359    private JMenuItem ivjExit = null;
360    private JButton ivjExitButton = null;
361    private JMenu ivjHelpMenu = null;
362    private JMenuItem ivjHelpTopics = null;
363    private JPanel ivjJAppletContentPane = null;
364    private JPanel ivjJPanel13 = null;
365    private JSeparator ivjJSeparator10 = null;
366    private JSeparator ivjJSeparator11 = null;
367    private JSeparator ivjJSeparator13 = null;
368    private JSeparator ivjJSeparator14 = null;
369    private JSeparator ivjJSeparator2 = null;
370    private JSeparator ivjJSeparator5 = null;
371    private JSeparator ivjJSeparator6 = null;
372    private JSeparator ivjJSeparator7 = null;
373    private JSeparator ivjJSeparator9 = null;
374    private JToolBar ivjJToolBar1 = null;
375    private JButton ivjJToolBarButton1 = null;

```

```

372     private JButton ivjJToolBarButton2 = null;
373     private JButton ivjJToolBarButton3 = null;
374     private JPanel ivjMessagePanel = null;
375     private BoxLayout ivjMessagePanelBoxLayout = null;
376     private ClientPanorama ivjPanoramaPanel = null;
377     private JMenuItem ivjRemoveDocuments = null;
378     private RemoveFiles ivjRemoveFiles1 = null;
379     private HelpDialog ivjHelpDialog1 = null;
380     private ClientManager clientManager;
381     private JProgressBar ivjMessageArea = null;
382     private JButton ivjJToolBarButton5 = null;
383     private JMenu ivjSetup = null;
384     private JButton ivjPreferencesButton = null;
385     private JMenuItem ivjPreferencesItem = null;
386     private PreferencesDialog ivjPreferencesDialog1 = null;
387     private JTree ivjDiscussionTree = null;
388     private JPanel ivjJPanel11 = null;
389     private JPanel ivjJPanel21 = null;
390     private JTextPane ivjMessageDisplayArea = null;
391     private JSplitPane ivjInnerSplitPane = null;
392     private JSplitPane ivjOuterSplitPane = null;
393     private JToolBar ivjJToolBar2 = null;
394     private JMenu ivjFileMenu = null;
395     private GroupDirectory ivjGroupDirectory1 = null;
396     private JMenuItem ivjGroups = null;
397     private JButton ivjComposeNewMessageButton = null;
398     private JButton ivjMessagePropertiesButton = null;
399     private JButton ivjDeleteButton = null;
400     private JScrollPane ivjScrollPane1 = null;
401     private boolean terminated = false;
402     private MessageWindow messageWindow = null;
403     private ZComboBox ivjDocumentBox = null;
404
405     class IvjEventHandler implements java.awt.event.ActionListener {
406         public void actionPerformed(java.awt.event.ActionEvent e) {
407             if (e.getSource() == ClientApplication.this.getAbout())
408                 connEtoM1(e);
409             if (e.getSource() == ClientApplication.this.getAddDocuments())
410                 connEtoM4(e);
411             if (e.getSource() == ClientApplication.this.getRemoveDocuments())
412                 connEtoM5(e);
413             if (e.getSource() == ClientApplication.this.getExit())
414                 connEtoM7(e);
415             if (e.getSource() == ClientApplication.this.getJToolBarButton3())
416                 connEtoM8(e);
417             if (e.getSource() == ClientApplication.this.getJToolBarButton1())
418                 connEtoM9(e);
419             if (e.getSource() == ClientApplication.this.getPreferencesItem())
420                 PreferencesMenuItemToDialog(e);
421             if (e.getSource() == ClientApplication.this.getPreferencesButton())
422                 PreferencesButtonToDialog(e);
423             if (e.getSource() == ClientApplication.this.getGroups())
424                 connEtoM2(e);
425             if (e.getSource() == ClientApplication.this.getJToolBarButton5())
426                 connEtoM3(e);
427             if (e.getSource() == ClientApplication.this.getExitButton())
428                 connEtoM6(e);
429             if (e.getSource() == ClientApplication.this.getExit())
430                 connEtoM10(e);
431         };
432     };
433     private java.util.Hashtable visibleDocumentsList = new Hashtable();
434     private javax.swing.Timer messageAreaTimer;
435     private boolean itemListenerActive = true;
436 /**
 * ClientApplication constructor comment.

```

```

438  /*
439   public ClientApplication() {
440     super();
441     initialize();
442   }
443   /**
444   * ClientApplication constructor comment.
445   * @param title java.lang.String
446   */
447   public ClientApplication(String title) {
448     super(title);
449   }
450   /**
451   * 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)
453   * @param senderName java.lang.String
454   * @param authorName java.lang.String
455   * @param documentName java.lang.String
456   */
457   public void addDocumentToVisibleList(String senderName, String authorName, String documentNameAndPath,
458   String documentName)
459   {
460     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName , documentNameAndPath );
461     if ( getVisibleDocumentsList().containsKey( senderName ) )
462     {
463       ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
464       docList.add( new DocumentItem( fqPath, documentName ) );
465     }
466     else
467     {
468       ArrayList docList = new ArrayList();
469       docList.add( new AuthorItem( authorName ) );
470       docList.add( new DocumentItem( fqPath, documentName ) );
471       getVisibleDocumentsList().put( senderName, docList );
472     }
473   }
474   /**
475   * Advance the progress bar at the bottom of the client application by one step.
476   * Creation date: (19-06-00 11:04:10)
477   */
478   public void advanceTask()
479   {
480     if ( ivjMessageArea.getValue() >= ivjMessageArea.getMaximum() )
481     {
482       initMessageAreaTimer();
483       updateMessageArea( ivjMessageArea.getValue() );
484     }
485     else
486     {
487       updateMessageArea( ivjMessageArea.getValue()+1 );
488     }
489   }
490   /**
491   * Experimental code to ensure that child windows such as dialog boxes are always on top. This didn't seem to work
492   * according to specifications, but I left the code here for future refinement.
493   * Creation date: (16-08-00 15:32:07)
494   */
495   public void bringChildWindowsOnTop()
496   {
497     for ( int i = 0; i < getOwnedWindows().length; i++ )
498     {
499       getOwnedWindows()[i].toFront();
500     }
501   }

```

```

502 * Center and size the window at predefined dimensions and position relative to the screen resolution
503 * Creation date: (05-08-00 19:20:24)
504 */
505 public void center()
506 {
507
508     Dimension size;
509     Dimension preferredSize = new Dimension(
510         Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
511                         getFRAME_HORIZONTAL_SIZE() )),
512         Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
513                         getFRAME_VERTICAL_SIZE() )));
513     Dimension defaultSize = new Dimension(
514         Integer.parseInt( ClientConstants.getDEFAULT_FRAME_HORIZONTAL_SIZE() ),
515         Integer.parseInt( ClientConstants.getDEFAULT_FRAME_VERTICAL_SIZE() ));
516     Point position;
517     Point preferredPosition = new Point(
518         Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
519                         getFRAME_X_COORDINATE() )),
520         Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
521                         getFRAME_Y_COORDINATE() )));
521     Point defaultPosition = new Point(
522         Integer.parseInt( ClientConstants.getDEFAULT_FRAME_X_COORDINATE() ),
523         Integer.parseInt( ClientConstants.getDEFAULT_FRAME_Y_COORDINATE() ));
524 // if no value has overridden the default setting then query the OS for the dimensions of the display
525 if ( preferredSize.equals( defaultSize ) )
526 {
527     size = new Dimension( (int) (getToolkit().getScreenSize().getWidth() * ClientConstants.
528                         getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
529                         (int) (getToolkit().getScreenSize().getHeight() * ClientConstants.
530                         getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) );
531 }
532 else
533 {
534     size = preferredSize;
535 }
536 if ( preferredPosition.equals( defaultPosition ) )
537 {
538     position = new Point( (int)( getToolkit().getScreenSize().getWidth() * ( (1.0 - ClientConstants.
539                         getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2 ) ),
540                         (int)( getToolkit().getScreenSize().getHeight() * ( (1.0 - ClientConstants.
541                         getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE())/2 ) ) );
542 }
543 else
544 {
545     position = preferredPosition;
546 }
547 // System.out.println( size.toString() );
548 setSize( size );
549 setLocation( position );
550 setState( java.awt.Frame.NORMAL );
551 repaint();
552 }
553 /**
554 * Check preconditions for opening a dialog box and, if fulfilled, open it, await completion and
555 * validation and then
556 * insert the new message, if any, into the topic tree where indicated by the user.
557 * Creation date: (10-07-00 22:00:41)
558 */
559 public void composeNewMessage()
560 {
561     if ( getClientManager().verifyGroupConnectionManagement() == false )
562     {
563         return;
564     }
565     if ( getDiscussionTree().getSelectionCount() == 0 )

```

```

560     {
561         ErrorBox noSelectionErrorBox = new ErrorBox();
562         noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
563         noSelectionErrorBox.show();
564         return;
565     }
566     else
567     {
568         ComposeNewMessage composeNewMessage = new ComposeNewMessage();
569         composeNewMessage.setClientApplication( this );
570         composeNewMessage.setAuthor( ClientInfo.getPreferences().getProperty( ClientConstants.
571             getAUTHOR_NAME() ) );
572         composeNewMessage.show();
573         if ( composeNewMessage.getMessage() != null )
574         {
575             // construct new message based on the user input in the dialog box and insert it into the topic
576             // tree.
577             Message message = composeNewMessage.getMessage();
578             message.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() ) );
579             message.setDocument( getClientManager().getClientInfo().getCurrentDocumentName() );
580
581             TreePath selectionPath = getDiscussionTree().getSelectionPath();
582             DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectionPath.
583                 getLastPathComponent();
584             DefaultMutableTreeNode newTreeNode = new DefaultMutableTreeNode( message.getMessageID(), false )
585                 ;
586             DefaultMutableTreeNode parentNode = null;
587             DefaultMutableTreeNode newNode = null;
588             int insertionIndex = 0;
589             if ( selectedNode.isLeaf() && ( selectedNode.getParent() != null ) )
590             {
591                 // create leaf node associated with new message
592                 parentNode = ( (DefaultMutableTreeNode) selectedNode.getParent() );
593                 newNode = newTreeNode;
594                 // insert node immediately below the selected leaf node
595                 insertionIndex = parentNode.getIndex( selectedNode )+ 1;
596             }
597             else
598             {
599                 newNode = new DefaultMutableTreeNode( message.getTitle(), true );
600                 newNode.add( newTreeNode );
601                 parentNode = selectedNode;
602                 // insert node below bottommost child of the selected node's parent
603                 insertionIndex = parentNode.getChildCount();
604             }
605             ( (DefaultTreeModel) getDiscussionTree().getModel() ).insertNodeInto( newNode, parentNode,
606                 parentNode.getChildCount() );
607             //getDiscussionTree().scrollPathToVisible( new TreePath( newNode.getPath() ) );
608             //getDiscussionTree().treeDidChange();
609             getDiscussionTree().repaint();
610
611             clientManager.sendMessage( message );
612             clientManager.sendTopicTreeUpdate( (DefaultTreeModel) getDiscussionTree().getModel() );
613         }
614     }
615     /**
616      * connEtoM1: (About.action.actionPerformed(java.awt.event.ActionEvent) --> AboutDialog1.show()V)
617      * @param arg1 java.awt.event.ActionEvent
618      */
619     /* WARNING: THIS METHOD WILL BE REGENERATED. */
620     private void connEtoM1(java.awt.event.ActionEvent arg1) {
621         try {
622             // user code begin {1}
623             // user code end

```

```

620     getAboutDialog1().show();
621     // user code begin {2}
622     // user code end
623 } catch (java.lang.Throwable ivjExc) {
624     // user code begin {3}
625     // user code end
626     handleException(ivjExc);
627 }
628 /**
629 * connEtoM10: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
630     terminateAndClose()V)
631 * @param arg1 java.awt.event.ActionEvent
632 */
633 /* WARNING: THIS METHOD WILL BE REGENERATED. */
634 private void connEtoM10(java.awt.event.ActionEvent arg1) {
635     try {
636         // user code begin {1}
637         // user code end
638         this.terminateAndClose();
639         // user code begin {2}
640         // user code end
641     } catch (java.lang.Throwable ivjExc) {
642         // user code begin {3}
643         // user code end
644         handleException(ivjExc);
645     }
646 }
647 /**
648 * connEtoM2: (Groups.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory1.show()V)
649 * @param arg1 java.awt.event.ActionEvent
650 */
651 /* WARNING: THIS METHOD WILL BE REGENERATED. */
652 private void connEtoM2(java.awt.event.ActionEvent arg1) {
653     try {
654         // user code begin {1}
655         getGroupDirectory1().updateTable();
656         // user code end
657         getGroupDirectory1().show();
658         // user code begin {2}
659         // user code end
660     } catch (java.lang.Throwable ivjExc) {
661         // user code begin {3}
662         // user code end
663         handleException(ivjExc);
664     }
665 }
666 /**
667 * Insert the method's description here.
668 * Creation date: (27-06-00 16:58:47)
669 * @param actionPerformed java.awt.event.ActionEvent
670 */
671 private void connEtoM20(ActionEvent actionPerformed)
672 {
673     try {
674         // user code begin {1}
675         // user code end
676         getPreferencesDialog1().show();
677         // user code begin {2}
678         // user code end
679     } catch (java.lang.Throwable ivjExc) {
680         // user code begin {3}
681         // user code end
682         handleException(ivjExc);
683     }
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 {
693         // user code begin {1}
694         // user code end
695         getGroupDirectory1().show();
696         // user code begin {2}
697         // user code end
698     } catch (java.lang.Throwable ivjExc) {
699         // user code begin {3}
700         // user code end
701         handleException(ivjExc);
702     }
703 }
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 {
711         // user code begin {1}
712         // user code end
713         getAddFiles1().show();
714         // user code begin {2}
715         // user code end
716     } catch (java.lang.Throwable ivjExc) {
717         // user code begin {3}
718         // user code end
719         handleException(ivjExc);
720     }
721 }
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 {
729         // user code begin {1}
730         // user code end
731         getRemoveFiles1().show();
732         // user code begin {2}
733         // user code end
734     } catch (java.lang.Throwable ivjExc) {
735         // user code begin {3}
736         // user code end
737         handleException(ivjExc);
738     }
739 }
740 /**
 * connEtoM6: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.
 *             terminateAndClose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
742 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM6(java.awt.event.ActionEvent arg1) {
743     try {

```

```

748     // user code begin {1}
749     // user code end
750     this.terminateAndClose();
751     // user code begin {2}
752     // user code end
753 } catch (java.lang.Throwable ivjExc) {
754     // user code begin {3}
755     // user code end
756     handleException(ivjExc);
757 }
758 /**
759 * connEtoM7: (Exit.action.actionPerformed(java.awt.event.ActionEvent) --> ClientApplication.dispose()V)
760 * @param arg1 java.awt.event.ActionEvent
761 */
762 /* WARNING: THIS METHOD WILL BE REGENERATED. */
763 private void connEtoM7(java.awt.event.ActionEvent arg1) {
764     try {
765         // user code begin {1}
766         prepareTermination();
767         // user code end
768         this.dispose();
769         // user code begin {2}
770         // user code end
771     } catch (java.lang.Throwable ivjExc) {
772         // user code begin {3}
773         // user code end
774         handleException(ivjExc);
775     }
776 }
777 /**
778 * connEtoM8: (JToolBarButton3.action.actionPerformed(java.awt.event.ActionEvent) --> AddFiles1.show()V)
779 * @param arg1 java.awt.event.ActionEvent
780 */
781 /* WARNING: THIS METHOD WILL BE REGENERATED. */
782 private void connEtoM8(java.awt.event.ActionEvent arg1) {
783     try {
784         // user code begin {1}
785         // user code end
786         getAddFiles1().show();
787         // user code begin {2}
788         // user code end
789     } catch (java.lang.Throwable ivjExc) {
790         // user code begin {3}
791         // user code end
792         handleException(ivjExc);
793     }
794 }
795 /**
796 * connEtoM9: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) --> RemoveFiles1.show
797 ()V)
798 * @param arg1 java.awt.event.ActionEvent
799 */
800 /* WARNING: THIS METHOD WILL BE REGENERATED. */
801 private void connEtoM9(java.awt.event.ActionEvent arg1) {
802     try {
803         // user code begin {1}
804         // user code end
805         getRemoveFiles1().show();
806         // user code begin {2}
807         // user code end
808     } catch (java.lang.Throwable ivjExc) {
809         // user code begin {3}
810         // user code end
811         handleException(ivjExc);
812     }

```

```

}
814 /**
 * Deletes a message from the topic tree if one is selected and has not been replied to.
815 * Creation date: (12-07-00 10:23:34)
 */
816 public void deleteMessage()
{
817     if ( getClientManager().verifyGroupConnectionManagement() == false )
818     {
819         return;
820     }
821     if ( getDiscussionTree().getSelectionCount() == 0 )
822     {
823         ErrorBox noSelectionErrorBox = new ErrorBox();
824         noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
825         noSelectionErrorBox.show();
826     }
827     else
828     {
829         TreePath selectedPath = getDiscussionTree().getSelectionPath();
830         DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) selectedPath.getLastPathComponent()
831             ;
832         // if the selected node is not a leaf, i.e. if it is an inner node, then exit immediately, without
833         // displaying an error message
834         if ( selectedNode.isLeaf() == false )
835         {
836             return;
837         }
838         DefaultMutableTreeNode parentNode = (DefaultMutableTreeNode) selectedNode.getParent();
839         DefaultMutableTreeNode successorNode = (DefaultMutableTreeNode) parentNode.getChildAtAfter(
840             selectedNode );
841         // if the selected node has not been responded to AND this client composed the message
842         if ( ( ( successorNode == null ) || ( successorNode.isLeaf() == false ) )&&
843             (getClientManager().getClientInfo().getCurrentMessage().getCreator().equalsIgnoreCase( (String)
844                 ) ClientInfo.getPreferences().get( ClientConstants.getNAME() ) ) )
845         )
846         {
847             ConfirmationBox confirmationBox = new ConfirmationBox();
848             confirmationBox.setText( ClientConstants.getCONFIRM_MESSAGE_DELETION( (String) getClientManager
849                 .getClientInfo().getCurrentMessage().getTitle() ) );
850             confirmationBox.show();
851             if ( confirmationBox.isConfirmed() )
852             {
853                 // user has confirmed deletion, so proceed accordingly
854                 clientManager.sendMessageDeletion( (String) selectedNode.getUserObject() );
855                 DefaultMutableTreeNode predecessorNode = (DefaultMutableTreeNode) parentNode.getChildAtBefore(
856                     selectedNode );
857                 // if selectedNode is the first and only child of an inner node, then remove that also
858                 if ( predecessorNode == null )
859                 {
860                     parentNode.removeFromParent();
861                 }
862                 selectedNode.removeFromParent();
863                 ( (DefaultTreeModel) getDiscussionTree().getModel() ).reload();
864                 clientManager.sendTopicTreeUpdate( (DefaultTreeModel) getDiscussionTree().getModel() );
865             }
866         }
867     }
868     else
869     {
870         ErrorBox invalidSelection = new ErrorBox();
871         invalidSelection.setText( ClientConstants.getINVALID_DELETION_ATTEMPTED_MESSAGE() );
872         invalidSelection.show();
873     }
874 }

```

```

    /**
874     * Insert the method's description here.
875     * Creation date: (28-07-00 00:27:12)
876     */
877     protected void finalize() throws Throwable
878     {
879         // System.out.println( " finalized " );
880         // ensure proper termination
881         /* if ( isTerminated() == false )
882         {
883             terminateAndClose();
884         }
885         */
886     }
887     /**
888      * Return the About property value.
889      * @return javax.swing.JMenuItem
890      */
891     /* WARNING: THIS METHOD WILL BE REGENERATED. */
892     private javax.swing.JMenuItem getAbout()
893     {
894         if (ivjAbout == null) {
895             try {
896                 ivjAbout = new javax.swing.JMenuItem();
897                 ivjAbout.setName("About");
898                 ivjAbout.setText("About");
899                 // user code begin {1}
900                 // user code end
901             } catch (java.lang.Throwable ivjExc) {
902                 // user code begin {2}
903                 // user code end
904                 handleException(ivjExc);
905             }
906         }
907         return ivjAbout;
908     }
909     /**
910      * Return the AboutDialog1 property value.
911      * @return clientapp.AboutDialog
912      */
913     /* WARNING: THIS METHOD WILL BE REGENERATED. */
914     private AboutDialog getAboutDialog1()
915     {
916         if (ivjAboutDialog1 == null) {
917             try {
918                 ivjAboutDialog1 = new peerviewmisc.AboutDialog();
919                 ivjAboutDialog1.setName("AboutDialog1");
920                 ivjAboutDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
921                 // user code begin {1}
922                 // user code end
923             } catch (java.lang.Throwable ivjExc) {
924                 // user code begin {2}
925                 // user code end
926                 handleException(ivjExc);
927             }
928         }
929         return ivjAboutDialog1;
930     }
931     /**
932      * Return the AddDocuments property value.
933      * @return javax.swing.JMenuItem
934      */
935     /* WARNING: THIS METHOD WILL BE REGENERATED. */
936     public javax.swing.JMenuItem getAddDocuments()
937     {
938         if (ivjAddDocuments == null) {
939             try {
940                 ivjAddDocuments = new javax.swing.JMenuItem();
941                 ivjAddDocuments.setName("AddDocuments");
942             }
943         }
944         return ivjAddDocuments;
945     }

```

```

940         ivjAddDocuments.setText("Add_Documents...");
941         // user code begin {1}
942         // user code end
943     } catch (java.lang.Throwable ivjExc) {
944         // user code begin {2}
945         // user code end
946         handleException(ivjExc);
947     }
948     return ivjAddDocuments;
949 }
950 /**
951 * Return the AddFiles1 property value.
952 * @return clientapp.AddFiles
953 */
954 /* WARNING: THIS METHOD WILL BE REGENERATED. */
955 private AddFiles getAddFiles1() {
956     if (ivjAddFiles1 == null) {
957         try {
958             ivjAddFiles1 = new clientapp.AddFiles();
959             ivjAddFiles1.setName("AddFiles1");
960             ivjAddFiles1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
961             // user code begin {1}
962             ivjAddFiles1.setClientApplication(this);
963             // user code end
964         } catch (java.lang.Throwable ivjExc) {
965             // user code begin {2}
966             // user code end
967             handleException(ivjExc);
968         }
969     }
970     return ivjAddFiles1;
971 }
972 /**
973 * Return the ClientApplicationJMenuBar property value.
974 * @return javax.swing.JMenuBar
975 */
976 /* WARNING: THIS METHOD WILL BE REGENERATED. */
977 private javax.swing.JMenuBar getClientApplicationJMenuBar() {
978     if (ivjClientApplicationJMenuBar == null) {
979         try {
980             ivjClientApplicationJMenuBar = new javax.swing.JMenuBar();
981             ivjClientApplicationJMenuBar.setName("ClientApplicationJMenuBar");
982             ivjClientApplicationJMenuBar.add(getFileMenu());
983             ivjClientApplicationJMenuBar.add(getSetup());
984             ivjClientApplicationJMenuBar.add(getHelpMenu());
985             // user code begin {1}
986             // user code end
987         } catch (java.lang.Throwable ivjExc) {
988             // user code begin {2}
989             // user code end
990             handleException(ivjExc);
991         }
992     }
993     return ivjClientApplicationJMenuBar;
994 }
995 /**
996 * Return the client manager object associated with this client and create it if it doesn't already
997 * exist.
998 * Creation date: (16-06-00 00:04:31)
999 * @return clientapp.ClientManager
1000 */
1001 public ClientManager getClientManager()
1002 {
1003     if (clientManager == null)
1004     {

```

```

1004     clientManager = new ClientManager(this);
1005   }
1006   return clientManager;
1007 }
1008 /**
1009  * Return the DefaultToolBarButton property value.
1010  * @return javax.swing.JButton
1011  */
1012 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1013 private javax.swing.JButton getComposeNewMessageButton() {
1014   if (ivjComposeNewMessageButton == null) {
1015     try {
1016       ivjComposeNewMessageButton = new javax.swing.JButton();
1017       ivjComposeNewMessageButton.setName("ComposeNewMessageButton");
1018       ivjComposeNewMessageButton.setToolTipText("Compose\u2014new\u2014message");
1019       ivjComposeNewMessageButton.setMnemonic('C');
1020       ivjComposeNewMessageButton.setText("");
1021       ivjComposeNewMessageButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1022       ivjComposeNewMessageButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1023       ivjComposeNewMessageButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
1024           toolbarButtonGraphics/general/ComposeMail16.gif")));
1025       ivjComposeNewMessageButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1026       // user code begin {1}
1027       ivjComposeNewMessageButton.addActionListener( new ComposeNewMessageButtonHandler() );
1028       // user code end
1029     } catch (java.lang.Throwable ivjExc) {
1030       // user code begin {2}
1031       // user code end
1032       handleException(ivjExc);
1033     }
1034   }
1035   return ivjComposeNewMessageButton;
1036 }
1037 /**
1038  * Return the DeleteButton property value.
1039  * @return javax.swing.JButton
1040  */
1041 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1042 private javax.swing.JButton getDeleteButton() {
1043   if (ivjDeleteButton == null) {
1044     try {
1045       ivjDeleteButton = new javax.swing.JButton();
1046       ivjDeleteButton.setName("DeleteButton");
1047       ivjDeleteButton.setToolTipText("Delete\u2014message");
1048       ivjDeleteButton.setMnemonic('D');
1049       ivjDeleteButton.setText("");
1050       ivjDeleteButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1051       ivjDeleteButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1052       ivjDeleteButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics
1053           /general/Delete16.gif")));
1054       ivjDeleteButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1055       // user code begin {1}
1056       ivjDeleteButton.addActionListener( new DeleteButtonHandler() );
1057       // user code end
1058     } catch (java.lang.Throwable ivjExc) {
1059       // user code begin {2}
1060       // user code end
1061       handleException(ivjExc);
1062     }
1063   }
1064   return ivjDeleteButton;
1065 }
1066 /**
1067  * Return the DiscussionTree property value. This method ought to be called getTopicTree() - TO DO
1068  * @return javax.swing.JTree
1069  */

```

```

1068 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1069 private javax.swing.JTree getDiscussionTree() {
1070     if (ivjDiscussionTree == null) {
1071         try {
1072             ivjDiscussionTree = new javax.swing.JTree();
1073             ivjDiscussionTree.setName("DiscussionTree");
1074             ivjDiscussionTree.setToolTipText("Discussion\u00e9tree");
1075             ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
1076             ivjDiscussionTree.setShowsRootHandles(true);
1077             ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 1));
1078             ivjDiscussionTree.setMinimumSize(new java.awt.Dimension(1, 1));
1079             // user code begin {1}
1080             DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode();
1081             ivjDiscussionTree.setModel( new DefaultTreeModel( rootNode ) );
1082             ivjDiscussionTree.setRootVisible( true );
1083             // ivjDiscussionTree.setCellRenderer( new DiscussionTreeCellRenderer() );
1084             ivjDiscussionTree.addMouseListener( new TopicTreeMouseAdapter() );
1085             ivjDiscussionTree.addTreeSelectionListener( new TopicTreeSelectionListener() );
1086             ivjDiscussionTree.getSelectionModel().setSelectionMode( TreeSelectionModel.SINGLE_TREE_SELECTION
1087                     );
1088             ivjDiscussionTree.setShowsRootHandles( true );
1089             // make sure that the root is expanded
1090             ivjDiscussionTree.expandRow(0);
1091             // user code end
1092         } catch (java.lang.Throwable ivjExc) {
1093             // user code begin {2}
1094             // user code end
1095             handleException(ivjExc);
1096         }
1097     }
1098     return ivjDiscussionTree;
1099 }
1100 /**
1101 * Return the DocumentBox property value.
1102 * @return javax.swing.JComboBox
1103 */
1104 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1105 private javax.swing.JComboBox getDocumentBox() {
1106     if (ivjDocumentBox == null) {
1107         try {
1108             ivjDocumentBox = new JComboBox();
1109             ivjDocumentBox.setName("DocumentBox");
1110             ivjDocumentBox.setToolTipText("Visible\u00e9documents");
1111             // user code begin {1}
1112             ivjDocumentBox.setEditable( false );
1113             ivjDocumentBox.addItemListener( new VisibleDocumentsBoxItemListener() );
1114             ivjDocumentBox.setRowCount( 15 );
1115             ivjDocumentBox.insertItemAt( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION()
1116                     ), 0 );
1117             ivjDocumentBox.setSelectedIndex( 0 );
1118             ivjDocumentBox.setBorder( BorderFactory.createEtchedBorder() );
1119             // user code end
1120         } catch (java.lang.Throwable ivjExc) {
1121             // user code begin {2}
1122             // user code end
1123             handleException(ivjExc);
1124         }
1125     }
1126     return ivjDocumentBox;
1127 }
1128 /**
1129 * Return the Exit property value.
1130 * @return javax.swing.JMenuItem
1131 */
1132 /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

1132 private javax.swing.JMenuItem getExit() {
1133     if (ivjExit == null) {
1134         try {
1135             ivjExit = new javax.swing.JMenuItem();
1136             ivjExit.setName("Exit");
1137             ivjExit.setText("Exit");
1138             // user code begin {1}
1139             // user code end
1140         } catch (java.lang.Throwable ivjExc) {
1141             // user code begin {2}
1142             // user code end
1143             handleException(ivjExc);
1144         }
1145     }
1146     return ivjExit;
1147 }
1148 /**
1149 * Return the ExitButton property value.
1150 * @return javax.swing.JButton
1151 */
1152 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1153 private javax.swing.JButton getExitButton() {
1154     if (ivjExitButton == null) {
1155         try {
1156             ivjExitButton = new javax.swing.JButton();
1157             ivjExitButton.setName("ExitButton");
1158             ivjExitButton.setToolTipText("Exit");
1159             ivjExitButton.setText("");
1160             ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1161             ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1162             ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
1163                 general/Stop24.gif")));
1164             ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1165             // user code begin {1}
1166             // user code end
1167         } catch (java.lang.Throwable ivjExc) {
1168             // user code begin {2}
1169             // user code end
1170             handleException(ivjExc);
1171         }
1172     }
1173     return ivjExitButton;
1174 }
1175 /**
1176 * Return the DocumentMenu property value.
1177 * @return javax.swing.JMenu
1178 */
1179 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1180 private javax.swing.JMenu getFileMenu() {
1181     if (ivjFileMenu == null) {
1182         try {
1183             ivjFileMenu = new javax.swing.JMenu();
1184             ivjFileMenu.setName("FileMenu");
1185             ivjFileMenu.setMnemonic('F');
1186             ivjFileMenu.setText("File");
1187             ivjFileMenu.add(getAddDocuments());
1188             ivjFileMenu.add(getRemoveDocuments());
1189             ivjFileMenu.add(getJSeparator14());
1190             ivjFileMenu.add(getExit());
1191             // user code begin {1}
1192             // user code end
1193         } catch (java.lang.Throwable ivjExc) {
1194             // user code begin {2}
1195             // user code end
1196             handleException(ivjExc);
1197         }

```

```

        }
1198     return ivjFileMenu;
}
1200 /**
 * Return the GroupDirectory1 property value.
1201 * @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) {
        // user code begin {2}
        // user code end
        handleException(ivjExc);
    }
1220     }
        return ivjGroupDirectory1;
}
1222 /**
 * 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) {
        try {
            ivjGroups = new javax.swing.JMenuItem();
        ivjGroups.setName("Groups");
        ivjGroups.setMnemonic('g');
1234         ivjGroups.setText("Groups");
            // user code begin {1}
1236         // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
1242     }
        return ivjGroups;
}
1244 /**
 * 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();
        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}
        // 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;
        try {
1282             {
1284                 String helpPathAndName = ClientConstants.getCLIENT_DOCUMENTATION_HELPSET_NAME();
                java.net.URL hsURL = HelpSet.findHelpSet( null, helpPathAndName );
                hs = new HelpSet( null, hsURL );
                HelpBroker hb = hs.createHelpBroker();
                getHelpTopics().addActionListener( new HelpActionListener( hb ) );
            }
1290             catch ( Exception E )
            {
                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 }
/**
 * 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();
        ivjHelpTopics.setName("HelpTopics");
        ivjHelpTopics.setText("HelpTopics");
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.
 * @return javax.swing.JSplitPane
 */
1326 /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

    private javax.swing.JSplitPane getInnerSplitPane() {
1330    if (ivjInnerSplitPane == null) {
1331        try {
1332            ivjInnerSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.HORIZONTAL_SPLIT);
1333            ivjInnerSplitPane.setName("InnerSplitPane");
1334            ivjInnerSplitPane.setDividerSize(6);
1335            ivjInnerSplitPane.setMaximumSize(new java.awt.Dimension(100000, 100000));
1336            ivjInnerSplitPane.setDividerLocation(300);
1337            ivjInnerSplitPane.setPreferredSize(new java.awt.Dimension(1, 1));
1338            ivjInnerSplitPane.setOneTouchExpandable(true);
1339            ivjInnerSplitPane.setBounds(224, 808, 798, 503);
1340            ivjInnerSplitPane.setMinimumSize(new java.awt.Dimension(1, 1));
1341            getInnerSplitPane().add(getJPanel11(), "left");
1342            getInnerSplitPane().add(getJPanel21(), "right");
1343            // user code begin {1}
1344            ivjInnerSplitPane.resetToPreferredSizes();
1345            // user code end
1346        } catch (java.lang.Throwable ivjExc) {
1347            // user code begin {2}
1348            // user code end
1349            handleException(ivjExc);
1350        }
1351    }
1352    return ivjInnerSplitPane;
1353 }
1354 /**
 * Return the JAppletContentPane property value.
1355 * @return javax.swing.JPanel
 */
1356 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJAppletContentPane() {
1357    if (ivjJAppletContentPane == null) {
1358        try {
1359            ivjJAppletContentPane = new javax.swing.JPanel();
1360            ivjJAppletContentPane.setName("JAppletContentPane");
1361            ivjJAppletContentPane.setLayout(new java.awt.BorderLayout());
1362            getJAppletContentPane().add(getMessagePanel(), "South");
1363            getJAppletContentPane().add(getJToolBar1(), "North");
1364            getJAppletContentPane().add(getOuterSplitPane(), "Center");
1365            // user code begin {1}
1366            // user code end
1367        } catch (java.lang.Throwable ivjExc) {
1368            // user code begin {2}
1369            // user code end
1370            handleException(ivjExc);
1371        }
1372    }
1373    return ivjJAppletContentPane;
1374 }
1375 /**
 * Return the JPanel11 property value.
1376 * @return javax.swing.JPanel
 */
1377 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJPanel11() {
1378    if (ivjJPanel11 == null) {
1379        try {
1380            ivjJPanel11 = new javax.swing.JPanel();
1381            ivjJPanel11.setName("JPanel11");
1382            ivjJPanel11.setPreferredSize(new java.awt.Dimension(150, 1));
1383            ivjJPanel11.setLayout(new java.awt.GridBagLayout());
1384            ivjJPanel11.setBackground(java.awt.SystemColor.window);
1385            ivjJPanel11.setMinimumSize(new java.awt.Dimension(1, 1));
1386
1387            java.awt.GridBagConstraints constraintsDiscussionTree = new java.awt.GridBagConstraints();
1388            constraintsDiscussionTree.gridx = 0; constraintsDiscussionTree.gridy = 0;
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
342
```

```

1396     constraintsDiscussionTree.fill = java.awt.GridBagConstraints.BOTH;
1397     constraintsDiscussionTree.anchor = java.awt.GridBagConstraints.WEST;
1398     constraintsDiscussionTree.weightx = 1.0;
1399     constraintsDiscussionTree.weighty = 1.0;
1400     getJPanel11().add(getDiscussionTree(), constraintsDiscussionTree);
1401
1402     java.awt.GridBagConstraints constraintsJToolBar2 = new java.awt.GridBagConstraints();
1403     constraintsJToolBar2.gridx = 1; constraintsJToolBar2.gridy = 0;
1404     constraintsJToolBar2.fill = java.awt.GridBagConstraints.VERTICAL;
1405     constraintsJToolBar2.anchor = java.awt.GridBagConstraints.NORTHEAST;
1406     getJPanel11().add(getJToolBar2(), constraintsJToolBar2);
1407     // user code begin {1}
1408     // user code end
1409 } catch (java.lang.Throwable ivjExc) {
1410     // user code begin {2}
1411     // user code end
1412     handleException(ivjExc);
1413 }
1414 return ivjJPanel11;
1415 }
1416 /**
1417 * Return the JPanel21 property value.
1418 * @return javax.swing.JPanel
1419 */
1420 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1421 private javax.swing.JPanel getJPanel21() {
1422     if (ivjJPanel21 == null) {
1423         try {
1424             ivjJPanel21 = new javax.swing.JPanel();
1425             ivjJPanel21.setName("JPanel21");
1426             ivjJPanel21.setPreferredSize(new java.awt.Dimension(450, 1));
1427             ivjJPanel21.setLayout(new java.awt.BorderLayout());
1428             ivjJPanel21.setMinimumSize(new java.awt.Dimension(1, 1));
1429             getJPanel21().add(getJScrollPane1(), "Center");
1430             // user code begin {1}
1431             // user code end
1432         } catch (java.lang.Throwable ivjExc) {
1433             // user code begin {2}
1434             // user code end
1435             handleException(ivjExc);
1436         }
1437     }
1438     return ivjJPanel21;
1439 }
1440 /**
1441 * Return the JPanel3 property value.
1442 * @return javax.swing.JPanel
1443 */
1444 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1445 private javax.swing.JPanel getJPanel3() {
1446     if (ivjJPanel3 == null) {
1447         try {
1448             ivjJPanel3 = new javax.swing.JPanel();
1449             ivjJPanel3.setName("JPanel3");
1450             ivjJPanel3.setLayout(new java.awt.GridBagLayout());
1451
1452             java.awt.GridBagConstraints constraintsJSeparator11 = new java.awt.GridBagConstraints();
1453             constraintsJSeparator11.gridx = 2; constraintsJSeparator11.gridy = 1;
1454             constraintsJSeparator11.gridwidth = 0;
1455             constraintsJSeparator11.ipadx = -1;
1456             constraintsJSeparator11.ipady = 1;
1457             constraintsJSeparator11.insets = new java.awt.Insets(19, 0, 20, 0);
1458             getJPanel3().add(getJSeparator11(), constraintsJSeparator11);
1459
1460             java.awt.GridBagConstraints constraintsJSeparator10 = new java.awt.GridBagConstraints();

```

```

1462     constraintsJSeparator10.gridx = 3; constraintsJSeparator10.gridy = 1;
1463     constraintsJSeparator10.gridwidth = 0;
1464     constraintsJSeparator10.ipadx = -1;
1465     constraintsJSeparator10.ipady = 1;
1466     constraintsJSeparator10.insets = new java.awt.Insets(19, 0, 20, 0);
1467     get JPanel3().add(getJSeparator10(), constraintsJSeparator10);

1468     java.awt.GridBagConstraints constraintsJToolBarButton3 = new java.awt.GridBagConstraints();
1469     constraintsJToolBarButton3.gridx = 3; constraintsJToolBarButton3.gridy = 1;
1470     constraintsJToolBarButton3.insets = new java.awt.Insets(5, 5, 5, 2);
1471     get JPanel3().add(getJToolBarButton3(), constraintsJToolBarButton3);
1472

1473     java.awt.GridBagConstraints constraintsJToolBarButton1 = new java.awt.GridBagConstraints();
1474     constraintsJToolBarButton1.gridx = 4; constraintsJToolBarButton1.gridy = 1;
1475     constraintsJToolBarButton1.insets = new java.awt.Insets(5, 3, 5, 5);
1476     get JPanel3().add(getJToolBarButton1(), constraintsJToolBarButton1);

1477     java.awt.GridBagConstraints constraintsJSeparator2 = new java.awt.GridBagConstraints();
1478     constraintsJSeparator2.gridx = 5; constraintsJSeparator2.gridy = 1;
1479     constraintsJSeparator2.gridwidth = 0;
1480     constraintsJSeparator2.ipadx = -1;
1481     constraintsJSeparator2.ipady = 1;
1482     constraintsJSeparator2.insets = new java.awt.Insets(19, 0, 20, 0);
1483     get JPanel3().add(getJSeparator2(), constraintsJSeparator2);

1484     java.awt.GridBagConstraints constraintsJSeparator5 = new java.awt.GridBagConstraints();
1485     constraintsJSeparator5.gridx = 6; constraintsJSeparator5.gridy = 1;
1486     constraintsJSeparator5.gridwidth = 0;
1487     constraintsJSeparator5.ipadx = -1;
1488     constraintsJSeparator5.ipady = 1;
1489     constraintsJSeparator5.insets = new java.awt.Insets(19, 0, 20, 0);
1490     get JPanel3().add(getJSeparator5(), constraintsJSeparator5);

1491     java.awt.GridBagConstraints constraintsJToolBarButton5 = new java.awt.GridBagConstraints();
1492     constraintsJToolBarButton5.gridx = 6; constraintsJToolBarButton5.gridy = 1;
1493     constraintsJToolBarButton5.insets = new java.awt.Insets(5, 5, 5, 2);
1494     get JPanel3().add(getJToolBarButton5(), constraintsJToolBarButton5);
1495

1496     java.awt.GridBagConstraints constraintsPreferencesButton = new java.awt.GridBagConstraints();
1497     constraintsPreferencesButton.gridx = 7; constraintsPreferencesButton.gridy = 1;
1498     constraintsPreferencesButton.insets = new java.awt.Insets(5, 3, 5, 5);
1499     get JPanel3().add(getPreferencesButton(), constraintsPreferencesButton);

1500     java.awt.GridBagConstraints constraintsJSeparator13 = new java.awt.GridBagConstraints();
1501     constraintsJSeparator13.gridx = 8; constraintsJSeparator13.gridy = 1;
1502     constraintsJSeparator13.gridwidth = 0;
1503     constraintsJSeparator13.ipadx = -1;
1504     constraintsJSeparator13.ipady = 1;
1505     constraintsJSeparator13.insets = new java.awt.Insets(19, 0, 20, 0);
1506     get JPanel3().add(getJSeparator13(), constraintsJSeparator13);

1507     java.awt.GridBagConstraints constraintsJSeparator7 = new java.awt.GridBagConstraints();
1508     constraintsJSeparator7.gridx = 9; constraintsJSeparator7.gridy = 1;
1509     constraintsJSeparator7.gridwidth = 0;
1510     constraintsJSeparator7.ipadx = -1;
1511     constraintsJSeparator7.ipady = 1;
1512     constraintsJSeparator7.insets = new java.awt.Insets(19, 0, 20, 0);
1513     get JPanel3().add(getJSeparator7(), constraintsJSeparator7);

1514     java.awt.GridBagConstraints constraintsJToolBarButton2 = new java.awt.GridBagConstraints();
1515     constraintsJToolBarButton2.gridx = 9; constraintsJToolBarButton2.gridy = 1;
1516     constraintsJToolBarButton2.insets = new java.awt.Insets(5, 5, 5, 2);
1517     get JPanel3().add(getJToolBarButton2(), constraintsJToolBarButton2);
1518

1519     java.awt.GridBagConstraints constraintsExitButton = new java.awt.GridBagConstraints();
1520     constraintsExitButton.gridx = 10; constraintsExitButton.gridy = 1;

```

```

1528     constraintsExitButton.insets = new java.awt.Insets(5, 3, 5, 5);
1529     getJPanel3().add(getExitButton(), constraintsExitButton);

1530     java.awt.GridBagConstraints constraintsJSeparator9 = new java.awt.GridBagConstraints();
1531     constraintsJSeparator9.gridx = 11; constraintsJSeparator9.gridy = 1;
1532     constraintsJSeparator9.gridwidth = 0;
1533     constraintsJSeparator9.ipadx = -1;
1534     constraintsJSeparator9.ipady = 1;
1535     constraintsJSeparator9.insets = new java.awt.Insets(19, 0, 20, 0);
1536     getJPanel3().add(getJSeparator9(), constraintsJSeparator9);

1537     java.awt.GridBagConstraints constraintsJSeparator6 = new java.awt.GridBagConstraints();
1538     constraintsJSeparator6.gridx = 12; constraintsJSeparator6.gridy = 1;
1539     constraintsJSeparator6.weightx = 1.0;
1540     constraintsJSeparator6.gridwidth = 0;
1541     constraintsJSeparator6.ipadx = -1;
1542     constraintsJSeparator6.ipady = 1;
1543     constraintsJSeparator6.insets = new java.awt.Insets(19, 0, 20, 0);
1544     getJPanel3().add(getJSeparator6(), constraintsJSeparator6);

1545     java.awt.GridBagConstraints constraintsDocumentBox = new java.awt.GridBagConstraints();
1546     constraintsDocumentBox.gridx = 13; constraintsDocumentBox.gridy = 1;
1547     constraintsDocumentBox.anchor = java.awt.GridBagConstraints.EAST;
1548     constraintsDocumentBox.weightx = 0.0;
1549     constraintsDocumentBox.weighty = 0.0;
1550     constraintsDocumentBox.ipadx = 250;
1551     constraintsDocumentBox.ipady = 10;
1552     constraintsDocumentBox.insets = new java.awt.Insets(7, 5, 7, 7);
1553     getJPanel3().add(getDocumentBox(), constraintsDocumentBox);
1554     // user code begin {1}
1555     // user code end
1556     } catch (java.lang.Throwable ivjExc) {
1557         // user code begin {2}
1558         // user code end
1559         handleException(ivjExc);
1560     }
1561 }
1562 return ivj JPanel3;
1563 }
1564 /**
1565 * Return the JScrollPane1 property value.
1566 * @return javax.swing.JScrollPane
1567 */
1568 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1569 private javax.swing.JScrollPane getJScrollPane1() {
1570     if (ivj JScrollPane1 == null) {
1571         try {
1572             ivj JScrollPane1 = new javax.swing.JScrollPane();
1573             ivj JScrollPane1.setName("JScrollPane1");
1574             getJScrollPane1().setViewportView(getMessageDisplayArea());
1575             // user code begin {1}
1576             // user code end
1577             } catch (java.lang.Throwable ivjExc) {
1578                 // user code begin {2}
1579                 // user code end
1580                 handleException(ivjExc);
1581             }
1582         }
1583     return ivj JScrollPane1;
1584 }
1585 /**
1586 * Return the JSeparator10 property value.
1587 * @return javax.swing.JSeparator
1588 */
1589 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1590 private javax.swing.JSeparator getJSeparator10() {

```

```

1594     if (ivjJSeparator10 == null) {
1595         try {
1596             ivjJSeparator10 = new javax.swing.JSeparator();
1597             ivjJSeparator10.setName("JSeparator10");
1598             // user code begin {1}
1599             // user code end
1600         } catch (java.lang.Throwable ivjExc) {
1601             // user code begin {2}
1602             // user code end
1603             handleException(ivjExc);
1604         }
1605     }
1606     return ivjJSeparator10;
1607 }
1608 /**
1609 * Return the JSeparator11 property value.
1610 * @return javax.swing.JSeparator
1611 */
1612 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1613 private javax.swing.JSeparator getJSeparator11() {
1614     if (ivjJSeparator11 == null) {
1615         try {
1616             ivjJSeparator11 = new javax.swing.JSeparator();
1617             ivjJSeparator11.setName("JSeparator11");
1618             // user code begin {1}
1619             // user code end
1620         } catch (java.lang.Throwable ivjExc) {
1621             // user code begin {2}
1622             // user code end
1623             handleException(ivjExc);
1624         }
1625     }
1626     return ivjJSeparator11;
1627 }
1628 /**
1629 * Return the JSeparator13 property value.
1630 * @return javax.swing.JSeparator
1631 */
1632 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1633 private javax.swing.JSeparator getJSeparator13() {
1634     if (ivjJSeparator13 == null) {
1635         try {
1636             ivjJSeparator13 = new javax.swing.JSeparator();
1637             ivjJSeparator13.setName("JSeparator13");
1638             ivjJSeparator13.setFont(new java.awt.Font("dialog", 0, 24));
1639             // user code begin {1}
1640             // user code end
1641         } catch (java.lang.Throwable ivjExc) {
1642             // user code begin {2}
1643             // user code end
1644             handleException(ivjExc);
1645         }
1646     }
1647     return ivjJSeparator13;
1648 }
1649 /**
1650 * Return the JSeparator14 property value.
1651 * @return javax.swing.JSeparator
1652 */
1653 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1654 private javax.swing.JSeparator getJSeparator14() {
1655     if (ivjJSeparator14 == null) {
1656         try {
1657             ivjJSeparator14 = new javax.swing.JSeparator();
1658             ivjJSeparator14.setName("JSeparator14");
1659             // user code begin {1}

```

```

1660         // user code end
1661     } catch (java.lang.Throwable ivjExc) {
1662         // user code begin {2}
1663         // user code end
1664         handleException(ivjExc);
1665     }
1666     return ivjJSeparator14;
1667 }
1668 /**
1669 * Return the JSeparator2 property value.
1670 * @return javax.swing.JSeparator
1671 */
1672 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1673 private javax.swing.JSeparator getJSeparator2() {
1674     if (ivjJSeparator2 == null) {
1675         try {
1676             ivjJSeparator2 = new javax.swing.JSeparator();
1677             ivjJSeparator2.setName("JSeparator2");
1678             // user code begin {1}
1679             // user code end
1680         } catch (java.lang.Throwable ivjExc) {
1681             // user code begin {2}
1682             // user code end
1683             handleException(ivjExc);
1684         }
1685     }
1686     return ivjJSeparator2;
1687 }
1688 /**
1689 * Return the JSeparator5 property value.
1690 * @return javax.swing.JSeparator
1691 */
1692 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1693 private javax.swing.JSeparator getJSeparator5() {
1694     if (ivjJSeparator5 == null) {
1695         try {
1696             ivjJSeparator5 = new javax.swing.JSeparator();
1697             ivjJSeparator5.setName("JSeparator5");
1698             // user code begin {1}
1699             // user code end
1700         } catch (java.lang.Throwable ivjExc) {
1701             // user code begin {2}
1702             // user code end
1703             handleException(ivjExc);
1704         }
1705     }
1706     return ivjJSeparator5;
1707 }
1708 /**
1709 * Return the JSeparator6 property value.
1710 * @return javax.swing.JSeparator
1711 */
1712 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1713 private javax.swing.JSeparator getJSeparator6() {
1714     if (ivjJSeparator6 == null) {
1715         try {
1716             ivjJSeparator6 = new javax.swing.JSeparator();
1717             ivjJSeparator6.setName("JSeparator6");
1718             // user code begin {1}
1719             // user code end
1720         } catch (java.lang.Throwable ivjExc) {
1721             // user code begin {2}
1722             // user code end
1723             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);
    }
}
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);
    }
}
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(get JPanel3(), 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);
    }
}
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) {
1797         try {
1798             ivjJToolBar2 = new javax.swing.JToolBar();
1799             ivjJToolBar2.setName("JToolBar2");
1800             ivjJToolBar2.setFloatable(false);
1801             ivjJToolBar2.setBackground(java.awt.Color.white);
1802             ivjJToolBar2.setOrientation(javax.swing.SwingConstants.VERTICAL);
1803             ivjJToolBar2.add(getComposeNewMessageButton());
1804             getJToolBar2().add(getDeleteButton(), getName());
1805             getJToolBar2().add(getMessagePropertiesButton(), getMessagePropertiesButton().getName());
1806             // user code begin {1}
1807             // user code end
1808         } catch (java.lang.Throwable ivjExc) {
1809             // user code begin {2}
1810             // user code end
1811             handleException(ivjExc);
1812         }
1813     }
1814     return ivjJToolBar2;
1815 }
1816 /**
    * Return the JToolBarButton1 property value.
1817 * @return javax.swing.JButton
    */
1818 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getJToolBarButton1() {
1822     if (ivjJToolBarButton1 == null) {
1823         try {
1824             ivjJToolBarButton1 = new javax.swing.JButton();
1825             ivjJToolBarButton1.setName("JToolBarButton1");
1826             ivjJToolBarButton1.setToolTipText("Remove Documents");
1827             ivjJToolBarButton1.setText("");
1828             ivjJToolBarButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1829             ivjJToolBarButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1830             ivjJToolBarButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource(
1831                 "toolbarButtonGraphics/general/Export24.gif")));
1832             ivjJToolBarButton1.setMargin(new java.awt.Insets(0, 0, 0, 0));
1833             // user code begin {1}
1834             // user code end
1835         } catch (java.lang.Throwable ivjExc) {
1836             // user code begin {2}
1837             // user code end
1838             handleException(ivjExc);
1839         }
1840     }
1841     return ivjJToolBarButton1;
1842 }
1843 /**
    * Return the JToolBarButton2 property value.
1844 * @return javax.swing.JButton
    */
1845 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getJToolBarButton2() {
1848     if (ivjJToolBarButton2 == null) {
1849         try {
1850             ivjJToolBarButton2 = new javax.swing.JButton();
1851             ivjJToolBarButton2.setName("JToolBarButton2");
1852             ivjJToolBarButton2.setToolTipText("Help");
1853             ivjJToolBarButton2.setText("");
1854             ivjJToolBarButton2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1855             ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

```

```

1856         ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
1857             toolbarButtonGraphics/general/Help24.gif")));
1858         ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
1859         // user code begin {1}
1860         // user code end
1861     } catch (java.lang.Throwable ivjExc) {
1862         // user code begin {2}
1863         // user code end
1864         handleException(ivjExc);
1865     }
1866     return ivjJToolBarButton2;
1867 }
1868 /**
1869 * Return the JToolBarButton3 property value.
1870 * @return javax.swing.JButton
1871 */
1872 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1873 private javax.swing.JButton getJToolBarButton3() {
1874     if (ivjJToolBarButton3 == null) {
1875         try {
1876             ivjJToolBarButton3 = new javax.swing.JButton();
1877             ivjJToolBarButton3.setName("JToolBarButton3");
1878             ivjJToolBarButton3.setToolTipText("Add_documents" );
1879             ivjJToolBarButton3.setText("");
1880             ivjJToolBarButton3.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1881             ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1882             ivjJToolBarButton3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
1883                 toolbarButtonGraphics/general/Import24.gif")));
1884             ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
1885             // user code begin {1}
1886             // user code end
1887         } catch (java.lang.Throwable ivjExc) {
1888             // user code begin {2}
1889             // user code end
1890             handleException(ivjExc);
1891         }
1892     }
1893     return ivjJToolBarButton3;
1894 }
1895 /**
1896 * Return the JToolBarButton5 property value.
1897 * @return javax.swing.JButton
1898 */
1899 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1900 private javax.swing.JButton getJToolBarButton5() {
1901     if (ivjJToolBarButton5 == null) {
1902         try {
1903             ivjJToolBarButton5 = new javax.swing.JButton();
1904             ivjJToolBarButton5.setName("JToolBarButton5");
1905             ivjJToolBarButton5.setText("");
1906             ivjJToolBarButton5.setToolTipText("Group_directory" );
1907             ivjJToolBarButton5.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
1908             ivjJToolBarButton5.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1909             ivjJToolBarButton5.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
1910                 toolbarButtonGraphics/general/Preferences24.gif")));
1911             ivjJToolBarButton5.setMargin(new java.awt.Insets(0, 0, 0, 0));
1912             // user code begin {1}
1913             // user code end
1914         } catch (java.lang.Throwable ivjExc) {
1915             // user code begin {2}
1916             // user code end
1917             handleException(ivjExc);
1918         }
1919     }
1920     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) {
1927         try {
1928             ivjMessageArea = new javax.swing.JProgressBar();
1929             ivjMessageArea.setName("MessageArea");
1930             ivjMessageArea.setString("Message area: double-click here to open in separate window...");
1931             ivjMessageArea.setStringPainted(true);
1932             // user code begin {1}
1933             ivjMessageArea.setPreferredSize( new Dimension( Integer.MAX_VALUE, (int) ivjMessageArea.
1934                 getPreferredSize().getHeight() ) );
1935             ivjMessageArea.setMaximumSize( ivjMessageArea.getPreferredSize() );
1936             ivjMessageArea.addMouseListener( new MessageAreaMouseListener() );
1937             // user code end
1938         } catch (java.lang.Throwable ivjExc) {
1939             // user code begin {2}
1940             // user code end
1941             handleException(ivjExc);
1942         }
1943     }
1944     return ivjMessageArea;
1945 }
1946 /**
 * Return the ButtonPanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
1948 /*
/* WARNING: THIS METHOD WILL BE REGENERATED. */
1950 private java.awt.FlowLayout getMessageAreaFlowLayout() {
1951     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
1952     try {
1953         /* Create part */
1954         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
1955         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
1956     } catch (java.lang.Throwable ivjExc) {
1957         handleException(ivjExc);
1958     };
1959     return ivjButtonPanelFlowLayout;
1960 }
1961 /**
 * 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() {
1967     return messageAreaTimer;
1968 }
1969 /**
 * Return the MessageDisplayArea property value.
 * @return javax.swing.JTextPane
 */
1972 /*
/* WARNING: THIS METHOD WILL BE REGENERATED. */
1974 private javax.swing.JTextPane getMessageDisplayArea() {
1975     if (ivjMessageDisplayArea == null) {
1976         try {
1977             ivjMessageDisplayArea = new javax.swing.JTextPane();
1978             ivjMessageDisplayArea.setName("MessageDisplayArea");
1979             ivjMessageDisplayArea.setToolTipText("null");
1980             ivjMessageDisplayArea.setPreferredSize(new java.awt.Dimension(7, 1));
1981             ivjMessageDisplayArea.setBounds(0, 0, 490, 498);
1982             ivjMessageDisplayArea.setEnabled(false);
1983             ivjMessageDisplayArea.setMinimumSize(new java.awt.Dimension(6, 1));

```

```

1984         // user code begin {1}
1985         ivjMessageDisplayArea.setToolTipText(null);
1986         // user code end
1987     } catch (java.lang.Throwable ivjExc) {
1988         // user code begin {2}
1989         // user code end
1990         handleException(ivjExc);
1991     }
1992 }
1993     return ivjMessageDisplayArea;
1994 }
1995 /**
1996 * Return the MessagePanel property value.
1997 * @return javax.swing.JPanel
1998 */
1999 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2000 private javax.swing.JPanel getMessagePanel() {
2001     if (ivjMessagePanel == null) {
2002         try {
2003             ivjMessagePanel = new javax.swing.JPanel();
2004             ivjMessagePanel.setName("MessagePanel");
2005             ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
2006
2007             getMessagePanel().add(getMessageArea(), getMessageArea().getName());
2008             // user code begin {1}
2009             // user code end
2010         } catch (java.lang.Throwable ivjExc) {
2011             // user code begin {2}
2012             // user code end
2013             handleException(ivjExc);
2014         }
2015     }
2016     return ivjMessagePanel;
2017 }
2018 /**
2019 * Return the MessagePanelBoxLayout property value.
2020 * @return javax.swing.BoxLayout
2021 */
2022 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2023 private javax.swing.BoxLayout getMessagePanelBoxLayout() {
2024     javax.swing.BoxLayout ivjMessagePanelBoxLayout = null;
2025     try {
2026         /* Create part */
2027         ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swing.BoxLayout.Y_AXIS);
2028     } catch (java.lang.Throwable ivjExc) {
2029         handleException(ivjExc);
2030     };
2031     return ivjMessagePanelBoxLayout;
2032 }
2033 /**
2034 * Return the JToolBarButton4 property value.
2035 * @return javax.swing.JButton
2036 */
2037 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2038 private javax.swing.JButton getMessagePropertiesButton() {
2039     if (ivjMessagePropertiesButton == null) {
2040         try {
2041             ivjMessagePropertiesButton = new javax.swing.JButton();
2042             ivjMessagePropertiesButton.setName("MessagePropertiesButton");
2043             ivjMessagePropertiesButton.setToolTipText("View_message_properties");
2044             ivjMessagePropertiesButton.setMnemonic('V');
2045             ivjMessagePropertiesButton.setText("");
2046             ivjMessagePropertiesButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
2047             ivjMessagePropertiesButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
2048         }
2049     }
2050     return ivjMessagePropertiesButton;
2051 }

```

```

2048         ivjMessagePropertiesButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
2049             toolbarButtonGraphics/general/Information16.gif")));
2050         ivjMessagePropertiesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
2051         // user code begin {1}
2052         ivjMessagePropertiesButton.addActionListener( new MessagePropertiesButtonHandler() );
2053         // user code end
2054     } catch (java.lang.Throwable ivjExc) {
2055         // user code begin {2}
2056         // user code end
2057         handleException(ivjExc);
2058     }
2059     return ivjMessagePropertiesButton;
2060 }
2061 /**
2062 * Insert the method's description here.
2063 * Creation date: (16-08-00 17:05:09)
2064 * @return clientapp.MessageWindow
2065 */
2066 public MessageWindow getMessageWindow()
2067 {
2068     if ( messageWindow == null )
2069     {
2070         messageWindow = new MessageWindow();
2071     }
2072     return messageWindow;
2073 }
2074 /**
2075 * Return the JSplitPane1 property value.
2076 * @return javax.swing.JSplitPane
2077 */
2078 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2079 private javax.swing.JSplitPane getOuterSplitPane() {
2080     if (ivjOuterSplitPane == null) {
2081         try {
2082             ivjOuterSplitPane = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
2083             ivjOuterSplitPane.setName("OuterSplitPane");
2084             ivjOuterSplitPane.setDividerSize(6);
2085             ivjOuterSplitPane.setLastDividerLocation(600);
2086             ivjOuterSplitPane.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
2087             ivjOuterSplitPane.setDividerLocation(600);
2088             ivjOuterSplitPane.setPreferredSize(new java.awt.Dimension(589, 1000));
2089             ivjOuterSplitPane.setAlignmentX(java.awt.Component.RIGHT_ALIGNMENT);
2090             ivjOuterSplitPane.setOneTouchExpandable(true);
2091             ivjOuterSplitPane.setMinimumSize(new java.awt.Dimension(503, 1000));
2092             getOuterSplitPane().add(getPanoramaPanel(), "top");
2093             // user code begin {1}
2094             ivjOuterSplitPane.setDividerLocation( 0.8 );
2095             // user code end
2096         } catch (java.lang.Throwable ivjExc) {
2097             // user code begin {2}
2098             // user code end
2099             handleException(ivjExc);
2100         }
2101     }
2102     return ivjOuterSplitPane;
2103 }
2104 /**
2105 * Insert the method's description here.
2106 * Creation date: (04-08-00 14:39:32)
2107 * @return clientapp.ClientPanorama
2108 */
2109 public ClientPanorama getPanoramaPanel() {
2110     if (ivjPanoramaPanel == null) {
2111         try {
2112             ivjPanoramaPanel = new clientapp.ClientPanorama();

```

```

        ivjPanoramaPanel.setName("PanoramaPanel");
2114    ivjPanoramaPanel.setToolTipText("null");
2115    ivjPanoramaPanel.setPreferredSize(new java.awt.Dimension(300, 300));
2116    ivjPanoramaPanel.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
2117    ivjPanoramaPanel.setMinimumSize(new java.awt.Dimension(1, 1));
2118    // user code begin {1}
2119    // user code end
2120    } catch (java.lang.Throwable ivjExc) {
2121        // user code begin {2}
2122        // user code end
2123        handleException(ivjExc);
2124    }
2125 }
2126 return ivjPanoramaPanel;
2127 */
2128 /**
2129 * Return the JToolBarButton4 property value.
2130 * @return javax.swing.JButton
2131 */
2132 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2133 private javax.swing.JButton getPreferencesButton() {
2134     if (ivjPreferencesButton == null) {
2135         try {
2136             ivjPreferencesButton = new javax.swing.JButton();
2137             ivjPreferencesButton.setName("PreferencesButton");
2138             ivjPreferencesButton.setText("");
2139             ivjPreferencesButton.setToolTipText("Preferences");
2140             ivjPreferencesButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
2141             ivjPreferencesButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
2142             ivjPreferencesButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/development/Host24.gif")));
2143             ivjPreferencesButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
2144             // user code begin {1}
2145             // user code end
2146         } catch (java.lang.Throwable ivjExc) {
2147             // user code begin {2}
2148             // user code end
2149             handleException(ivjExc);
2150         }
2151     }
2152     return ivjPreferencesButton;
2153 }
2154 /**
2155 * Return the PreferencesDialog1 property value.
2156 * @return clientapp.PreferencesDialog
2157 */
2158 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2159 private PreferencesDialog getPreferencesDialog1() {
2160     if (ivjPreferencesDialog1 == null) {
2161         try {
2162             ivjPreferencesDialog1 = new clientapp.PreferencesDialog();
2163             ivjPreferencesDialog1.setName("PreferencesDialog1");
2164             ivjPreferencesDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
2165             // user code begin {1}
2166             ivjPreferencesDialog1.setClientApplication( this );
2167             ivjPreferencesDialog1.init();
2168             // user code end
2169         } catch (java.lang.Throwable ivjExc) {
2170             // user code begin {2}
2171             // user code end
2172             handleException(ivjExc);
2173         }
2174     }
2175     return ivjPreferencesDialog1;
2176 }
2177 */

```

```

2178 * Return the OptionsItem property value.
2179 * @return javax.swing.JMenuItem
2180 */
2181 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2182 private javax.swing.JMenuItem getPreferencesItem() {
2183     if (ivjPreferencesItem == null) {
2184         try {
2185             ivjPreferencesItem = new javax.swing.JMenuItem();
2186             ivjPreferencesItem.setName("PreferencesItem");
2187             ivjPreferencesItem.setText("Preferences");
2188             // user code begin {1}
2189             // user code end
2190         } catch (java.lang.Throwable ivjExc) {
2191             // user code begin {2}
2192             // user code end
2193             handleException(ivjExc);
2194         }
2195     }
2196     return ivjPreferencesItem;
2197 }
2198 /**
2199 * Return the RemoveDocuments property value.
2200 * @return javax.swing.JMenuItem
2201 */
2202 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2203 public javax.swing.JMenuItem getRemoveDocuments() {
2204     if (ivjRemoveDocuments == null) {
2205         try {
2206             ivjRemoveDocuments = new javax.swing.JMenuItem();
2207             ivjRemoveDocuments.setName("RemoveDocuments");
2208             ivjRemoveDocuments.setText("RemoveDocuments...");
2209             // user code begin {1}
2210             // user code end
2211         } catch (java.lang.Throwable ivjExc) {
2212             // user code begin {2}
2213             // user code end
2214             handleException(ivjExc);
2215         }
2216     }
2217     return ivjRemoveDocuments;
2218 }
2219 /**
2220 * Return the RemoveFiles1 property value.
2221 * @return clientapp.RemoveFiles
2222 */
2223 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2224 private RemoveFiles getRemoveFiles1() {
2225     if (ivjRemoveFiles1 == null) {
2226         try {
2227             ivjRemoveFiles1 = new clientapp.RemoveFiles();
2228             ivjRemoveFiles1.setName("RemoveFiles1");
2229             // user code begin {1}
2230             ivjRemoveFiles1.setClientApplication( this );
2231             ivjRemoveFiles1.initialize();
2232             // user code end
2233         } catch (java.lang.Throwable ivjExc) {
2234             // user code begin {2}
2235             // user code end
2236             handleException(ivjExc);
2237         }
2238     }
2239     return ivjRemoveFiles1;
2240 }
2241 /**
2242 * Return the Setup property value.
2243 * @return javax.swing.JMenu

```

```

2244  /*
2245  * WARNING: THIS METHOD WILL BE REGENERATED.
2246  */
2247  private javax.swing.JMenu getSetup() {
2248      if (ivjSetup == null) {
2249          try {
2250              ivjSetup = new javax.swing.JMenu();
2251              ivjSetup.setName("Setup");
2252              ivjSetup.setMnemonic('S');
2253              ivjSetup.setText("Setup");
2254              ivjSetup.add(getGroups());
2255              ivjSetup.add(getPreferencesItem());
2256              // user code begin {1}
2257              // user code end
2258          } catch (java.lang.Throwable ivjExc) {
2259              // user code begin {2}
2260              // user code end
2261              handleException(ivjExc);
2262          }
2263      }
2264      return ivjSetup;
2265  }
2266 /**
2267 * Return the visibleDocumentsList property.
2268 * Creation date: (30-08-00 16:14:33)
2269 * @return java.util.Hashtable
2270 */
2271 public java.util.Hashtable getVisibleDocumentsList() {
2272     return visibleDocumentsList;
2273 }
2274 /**
2275 * Called whenever the part throws an exception.
2276 * @param exception java.lang.Throwable
2277 */
2278 private void handleException(java.lang.Throwable exception) {
2279     /* Uncomment the following lines to print uncaught exceptions to stdout */
2280     System.out.println("-----\u2014UNCAUGHT\u2014EXCEPTION\u2014-----");
2281     exception.printStackTrace(System.out);
2282 }
2283 /**
2284 * Insert the method's description here.
2285 * Creation date: (11-08-00 17:57:58)
2286 * @param mouseEvent java.awt.event.MouseEvent
2287 */
2288 public void handleTopicTreeClick( TreePath selPath )
2289 {
2290     if ( selPath != null )
2291     {
2292         DefaultMutableTreeNode treeNode = (DefaultMutableTreeNode) selPath.getLastPathComponent();
2293         if ( treeNode.isLeaf() == true && treeNode.getUserObject() != null )
2294         {
2295             // retrieve ID for the message object associated with the selected node and request it from the
2296             // server
2297             String messageID = (String) treeNode.getUserObject();
2298             getClientManager().requestMessageFromServer( messageID );
2299             // System.out.println( "Sendte bud efter " + messageID );
2300         }
2301         else
2302         {
2303             setMessage( null );
2304             getClientManager().getClientInfo().setCurrentMessage( null );
2305         }
2306     }
2307 }
2308 /**

```

```

2310 * Initializes connections
2310 * @exception java.lang.Exception The exception description.
2310 */
2312 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2312 private void initConnections() throws java.lang.Exception {
2314     // user code begin {1}
2314     // user code end
2316     getAbout().addActionListener(ivjEventHandler);
2316     getAddDocuments().addActionListener(ivjEventHandler);
2318     getRemoveDocuments().addActionListener(ivjEventHandler);
2318     getExit().addActionListener(ivjEventHandler);
2320     getJToolBarButton3().addActionListener(ivjEventHandler);
2320     getJToolBarButton1().addActionListener(ivjEventHandler);
2322     getHelpTopics().addActionListener(ivjEventHandler);
2322     getJToolBarButton2().addActionListener(ivjEventHandler);
2324     getPreferencesItem().addActionListener(ivjEventHandler);
2324     getPreferencesButton().addActionListener(ivjEventHandler);
2326     getGroups().addActionListener(ivjEventHandler);
2326     getJToolBarButton5().addActionListener(ivjEventHandler);
2328     getExitButton().addActionListener(ivjEventHandler);
2328 }
2330 /**
2330 * Initialize the class.
2330 */
2332 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2334 private void initialize()
2334 {
2336     try {
2336         // user code begin {1}
2338         initPanoramaPanel();
2338         // user code end
2340         setName("ClientApplication");
2340         setTitle("PeerView\u00d7client");
2342         setSize(800, 600);
2342         setJMenuBar getClientApplicationJMenuBar());
2344         setContentPane getJAppletContentPane());
2344         initConnections();
2346     } catch (java.lang.Throwable ivjExc) {
2346         handleException(ivjExc);
2348     }
2348     // user code begin {2}
2350     setDefaultCloseOperation( WindowConstants.DO_NOTHING_ON_CLOSE );
2350     addWindowListener( new ClientWindowListener() );
2352     getMessageWindow().setVisible( false );
2352     getPanoramaPanel().setOwner(this);
2354     resetMessageArea();
2354     // the below code is inserted here due to difficulty using the visual composer to properly display
2354     // the top and
2356     // bottom components of the outer split pane.
2356     getOuterSplitPane().add( getInnerSplitPane(), "bottom");
2358     center();
2358     getClientManager().configure();
2360
2362     // user code end
2362
2364 }
2364 /**
2366 * Insert the method's description here.
2366 * Creation date: (18-06-00 17:16:36)
2368 * @param minimum int
2368 * @param maximum int
2370 * @param value int
2370 * @param string java.lang.String
2372 */
2372 public void initMessageArea(int minimum, int maximum, int value, String string)

```

```

2374 {
2375     ivjMessageArea.setMinimum( minimum );
2376     ivjMessageArea.setMaximum( maximum );
2377     ivjMessageArea.setValue( value );
2378     ivjMessageArea.setString( string );
2379     getMessageWindow().addString( string );
2380     // make sure the message disappears after a set interval if the progress bar is fixed
2381     if ( minimum == maximum )
2382     {
2383         initMessageAreaTimer();
2384     }
2385     // turn off any ticking timers so that update is not interrupted
2386     else
2387     {
2388         if ( getMessageAreaTimer() != null )
2389         {
2390             getMessageAreaTimer().stop();
2391         }
2392     }
2393 }
2394 /**
2395 * Initialize the message area timer.
2396 * Creation date: (08-09-00 23:53:48)
2397 */
2398 private void initMessageAreaTimer()
2399 {
2400     // start a timer to allow the current message a brief display interval before being deleted.
2401     if ( ( getMessageAreaTimer() == null ) ||
2402         ( getMessageAreaTimer().isRunning() == false ) )
2403     {
2404         setMessageAreaTimer( new Timer
2405             ( ClientConstants.getRESET_MESSAGE_AREA_DELAY(),
2406                 new ActionListener()
2407                 {
2408                     public void actionPerformed( ActionEvent ae )
2409                     {
2410                         resetMessageArea();
2411                     }
2412                 }
2413             ) );
2414         getMessageAreaTimer().setRepeats( false );
2415         getMessageAreaTimer().start();
2416     }
2417 }
2418 /**
2419 * Return the PanoramaPanel property value.
2420 * @return clientapp.ClientPanorama
2421 */
2422 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2423 public ClientPanorama initPanoramaPanel()
2424 {
2425     if ( ivjPanoramaPanel == null ){
2426         try {
2427             ivjPanoramaPanel = new clientapp.ClientPanorama();
2428             ivjPanoramaPanel.setName("PanoramaPanel");
2429             ivjPanoramaPanel.setPreferredSize(new java.awt.Dimension(300, 300));
2430             ivjPanoramaPanel.setAlignmentY(java.awt.Component.BOTTOM_ALIGNMENT);
2431             ivjPanoramaPanel.setMinimumSize(new java.awt.Dimension(1, 1));
2432             // user code begin {1}
2433             // user code end
2434         } catch (java.lang.Throwable ivjExc) {
2435             // user code begin {2}
2436             // user code end
2437             handleException(ivjExc);
2438         }
2439     }

```

```

2440     return ivjPanoramaPanel;
2441 }
2442 /**
2443 * Insert the method's description here.
2444 * Creation date: (05-09-00 22:15:03)
2445 * @return boolean
2446 */
2447 public boolean isItemListenerActive() {
2448     return itemListenerActive;
2449 }
2450 /**
2451 * Insert the method's description here.
2452 * Creation date: (15-08-00 08:48:45)
2453 * @return boolean
2454 */
2455 public boolean isTerminated() {
2456     return terminated;
2457 }
2458 /**
2459 * Lock interface by disabling those menu items that give access to activities that run in separate
2460 * threads.
2461 * The purpose of this is to minimize the potential for thread conflicts when operations such as
2462 * adddocuments and
2463 * removedocuments are set to execute in their own threads.
2464 * Creation date: (09-10-00 21:49:32)
2465 * @param value boolean
2466 */
2467 public void lockInterface(boolean value)
2468 {
2469     getAddDocuments().setEnabled( !value );
2470     getRemoveDocuments().setEnabled( !value );
2471     getGroups().setEnabled( !value );
2472     getPanoramaPanel().setPanoramaLock( value );
2473 }
2474 /**
2475 * main entrypoint - starts the part when it is run as an application
2476 * @param args java.lang.String[]
2477 */
2478 public static void main(java.lang.String[] args)
2479 {
2480     try {
2481         ClientApplication aClientApplication;
2482         aClientApplication = new ClientApplication();
2483         aClientApplication.addWindowListener(new java.awt.event.WindowAdapter() {
2484             public void windowClosing(java.awt.event.WindowEvent e) {
2485                 System.exit(0);
2486             };
2487         });
2488         aClientApplication.setVisible(true);
2489     } catch (Throwable exception) {
2490         System.err.println("Exception occurred in main() of javax.swing.JFrame");
2491         exception.printStackTrace(System.out);
2492     }
2493 }
2494 /**
2495 * PreferencesButtonToDialog: (PreferencesButton.action.actionPerformed(java.awt.event.ActionEvent) -->
2496 * PreferencesDialog1.show())
2497 * @param arg1 java.awt.event.ActionEvent
2498 */
2499 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2500 private void PreferencesButtonToDialog(java.awt.event.ActionEvent arg1) {
2501     try {
2502         // user code begin {1}
2503         // user code end
2504         getPreferencesDialog1().show();

```

```

2504     // user code begin {2}
2505     // user code end
2506 } catch (java.lang.Throwable ivjExc) {
2507     // user code begin {3}
2508     // user code end
2509     handleException(ivjExc);
2510 }
2511 /**
2512 * connEtoM6: (PreferencesItem.action.actionPerformed(java.awt.event.ActionEvent) --> PreferencesDialog1
2513 .show()V)
2514 */
2515 /* WARNING: THIS METHOD WILL BE REGENERATED. */
2516 private void PreferencesMenuItemToDialog(java.awt.event.ActionEvent arg1) {
2517     try {
2518         // user code begin {1}
2519         // user code end
2520         getPreferencesDialog1().show();
2521         // user code begin {2}
2522         // user code end
2523     } catch (java.lang.Throwable ivjExc) {
2524         // user code begin {3}
2525         // user code end
2526         handleException(ivjExc);
2527     }
2528 }
2529 /**
2530 * Prepare termination of the object.
2531 * Creation date: (28-07-00 00:34:11)
2532 */
2533 public void prepareTermination()
2534 {
2535     if ( getClientManager().getClientInfo().isLocalMode() == false )
2536     {
2537         getClientManager().disconnectFromGroup( getClientManager().getClientInfo().getActiveGroup() );
2538     }
2539     getClientManager().informServerOfExit();
2540     getClientManager().getClientInfo().writePreferences();
2541     setTerminated( true );
2542 }
2543 /**
2544 * Remove a document from the visible document list box.
2545 * Creation date: (30-08-00 15:43:31)
2546 * @param senderName java.lang.String
2547 * @param documentName java.lang.String
2548 */
2549 public void removeDocumentFromVisibleList(String senderName, String documentName)
2550 {
2551     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
2552     if ( getVisibleDocumentsList().containsKey( senderName ) )
2553     {
2554         ArrayList docList = (ArrayList) getVisibleDocumentsList().get( senderName );
2555         for ( int i = 0; i < docList.size(); i++ )
2556         {
2557             if ( docList.get( i ) instanceof DocumentItem )
2558             {
2559                 if ( ( DocumentItem ) docList.get( i ) ).getDocumentID().equals( fqPath ) )
2560                 {
2561                     docList.remove( i );
2562                 }
2563             }
2564         }
2565         // if group member no longer contributes to panorama, i.e. if the author's name is the only entry
2566         // ,then remove the list
2567         if ( docList.size() == 1 )

```

```

2568     {
2569         getVisibleDocumentsList().remove( senderName );
2570     }
2570 }
2572 /**
2573 * Clear the message area of the client application main window, i.e. empty it so that it appears blank
2574 * to the user.
2574 * Creation date: (18-06-00 17:17:37)
2575 */
2576 public void resetMessageArea()
2577 {
2578     getMessageArea().setSize( getMessagePanel().getSize() );
2579     getMessageArea().setString(ClientConstants.getMESSAGE_AREA_DEFAULT_STRING());
2580     getMessageArea().setValue(0);
2581     validate();
2582     repaint();
2583 }
2584 /**
2585 * Set method for the client mananager object.
2586 * Creation date: (16-06-00 00:04:31)
2587 * @param newClientManager clientapp.ClientManager
2588 */
2589 public void setClientManager(ClientManager newClientManager) {
2590     clientManager = newClientManager;
2591 }
2592 /**
2593 * Set method for the itemListenerActive field.
2594 * Creation date: (05-09-00 22:15:03)
2595 * @param newItemListenerActive boolean
2596 */
2597 public void setItemListenerActive(boolean newItemListenerActive) {
2598     itemListenerActive = newItemListenerActive;
2599 }
2600 /**
2601 * Set the message display area parameters to reflect the contents of the argument or reset to default
2602 * values if argument is null.
2603 * Creation date: (10-07-00 14:44:52)
2604 * @param newMessage java.lang.String
2605 */
2606 public void setMessage( Message message )
2607 {
2608     if ( message != null )
2609     {
2610         getMessageDisplayArea().setText( message.getContents() );
2611         getMessageDisplayArea().setToolTipText( message.toString() );
2612     }
2613     else
2614     {
2615         getMessageDisplayArea().setText( "" );
2616         getMessageDisplayArea().setToolTipText( null );
2617     }
2618     getMessageDisplayArea().repaint();
2619 }
2620 /**
2621 * Set method for the messageAreaTimer field.
2622 * Creation date: (31-08-00 02:18:56)
2623 * @param newMessageAreaTimer javax.swing.Timer
2624 */
2625 public void setMessageAreaTimer(javax.swing.Timer newMessageAreaTimer) {
2626     messageAreaTimer = newMessageAreaTimer;
2627 }
2628 /**
2629 * Set method for the messsageWindow field.
2630 * Creation date: (16-08-00 17:05:09)

```

```

2632 * @param newMessageWindow clientapp.MessageWindow
2632 */
2633 public void setMessageWindow(MessageWindow newMessageWindow) {
2634     messageWindow = newMessageWindow;
2635 }
2636 /**
2637 * Set method for the terminated field.
2638 * Creation date: (15-08-00 08:48:45)
2639 * @param newTerminated boolean
2640 */
2641 public void setTerminated(boolean newTerminated) {
2642     terminated = newTerminated;
2643 }
2644 /**
2645 * Set method for the discussion tree field.
2646 * Creation date: (09-07-00 23:03:10)
2647 * @param newTree DefaultTreeModel
2648 */
2649 public void setTopicTree(javax.swing.tree.DefaultTreeModel newTree)
2650 {
2651     // reset selection to top row to avoid triggering the change listener and have a spurious message
2652     // inserted into the message area
2653     if ( getDiscussionTree().isSelectionEmpty() == false )
2654     {
2655         getDiscussionTree().setSelectionRow( 0 );
2656     }
2657     getDiscussionTree().setModel( newTree );
2658     getDiscussionTree().revalidate();
2659     getDiscussionTree().repaint();
2660     // System.out.println( "Inde i setTopicTree" );
2661 }
2662 /**
2663 * Set visible documents list.
2664 * Creation date: (30-08-00 16:14:33)
2665 * @param newVisibleDocumentsList java.util.Hashtable
2666 */
2667 public void setVisibleDocumentsList(java.util.Hashtable newVisibleDocumentsList)
2668 {
2669     visibleDocumentsList = newVisibleDocumentsList;
2670 }
2671 /**
2672 * Initialize the message properties box with the contents of the current message, if any.
2673 * Creation date: (19-07-00 20:11:38)
2674 */
2675 public void showMessageProperties()
2676 {
2677     if ( getDiscussionTree().getSelectionCount() == 0 )
2678     {
2679         ErrorBox noSelectionErrorBox = new ErrorBox();
2680         noSelectionErrorBox.setText( ClientConstants.getNO_TOPIC_TREE_SELECTION_ERROR_MESSAGE() );
2681         noSelectionErrorBox.show();
2682         return;
2683     }
2684     else if ( getClientManager().verifyGroupConnectionManagement() == false )
2685     {
2686         return;
2687     }
2688     else if ( getClientManager().getClientInfo().getCurrentMessage() != null )
2689     {
2690         MessagePropertiesBox messagePropertiesBox = new MessagePropertiesBox();
2691         messagePropertiesBox.initTable( getClientManager().getClientInfo().getCurrentMessage() );
2692         messagePropertiesBox.show();
2693     }
2694 }
2695 /**

```

```

    * 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   */
2699   public void showMessageWindow()
2700   {
2701     getMessageWindow().setVisible( true );
2702   }
2703   /**
2704   * Prepare termination of the client application and then dispose of it.
    * Creation date: (12-09-00 21:19:40)
2706   */
2707   public void terminateAndClose()
2708   {
2709     prepareTermination();
2710     getClientManager().prepareForTermination();
2711     this.dispose();
2712   }
2713   /**
2714   * Test method. Purely for development purposes. Will be removed in stable releases.
    * Creation date: (29-06-00 18:52:19)
2716   */
2717   public void test()
2718   {
2719     PeerViewClient client = new PeerViewClient("PeerViewClient");
2720     // DefaultClient client = new DefaultClient("WTEST");
2721
2722     try
2723     {
2724       urlString = urlString.createSessionURL("localhost", 4461 , "socket", "test");
2725       if (SessionFactory.sessionExists( urlString ))
2726       {
2727         // System.out.println( "Session findes" );
2728       }
2729       Session session = SessionFactory.createSession(client, urlString, true );
2730       if ( session.channelExists( "test" ) )
2731       {
2732         // System.out.println( "channelExists" );
2733         Channel channel = session.createChannel(client, "test", true, true, true);
2734
2735         channel.addConsumer(client,client);
2736       }
2737     }
2738     catch ( JSDTEException JSDTE )
2739     {
2740       getClientManager().handleJSDTEException( JSDTE );
2741     }
2742   }
2743   /**
2744   * Change value field of the progress/message bar at the bottom of the client application main window.
    * Creation date: (18-06-00 17:17:07)
    * @param newValue int
2746   */
2747   public void updateMessageArea(int newValue)
2748   {
2749     ivjMessageArea.getModel().setValue(newValue);
2750     ivjMessageArea.repaint();
2751     // ivjMessageArea.paintImmediately(ivjMessageArea.getBounds());
2752   }
2753   /**
2754   * Convenience method for updating the visible documents box.
    * Creation date: (30-08-00 22:30:55)
2756   */
2757   public void updateVisibleDocumentsBox()

```

```

2760 {
2761     updateVisibleDocumentsBox( getVisibleDocumentsList() );
2762 }
2763 /**
2764 * Update contents of visible documents box.
2765 * Creation date: (30-08-00 01:03:02)
2766 * @param documents java.util.Hashtable
2767 */
2768 public void updateVisibleDocumentsBox(Hashtable documents)
2769 {
2770     // temporarily disable itemlistener to avoid repeated triggering
2771     // the below ought to be substitutable with a call to removeAllItem but an attempt to do so seemed to
2772     // fail.
2773     setItemListenerActive( false );
2774     /* while ( getDocumentBox().getItemCount() > 0 )
2775     {
2776         getDocumentBox().removeItemAt( 0 );
2777     }
2778 */
2779     getDocumentBox().setModel( new DefaultComboBoxModel() );
2780     getDocumentBox().insertItemAt( new CaptionItem( ClientConstants.getVISIBLE_DOCUMENTS_BOX_CAPTION() )
2781         , 0 );
2782     Collection values = documents.values();
2783     Iterator iterator = values.iterator();
2784     while ( iterator.hasNext() )
2785     {
2786         ArrayList itemList = (ArrayList) iterator.next();
2787         Iterator itemListIterator = itemList.iterator();
2788         while ( itemListIterator.hasNext() )
2789         {
2790             Object item = itemListIterator.next();
2791             getDocumentBox().addItem( item );
2792             // System.out.println( item.toString() );
2793         }
2794     }
2795     setItemListenerActive( true );
2796     getDocumentBox().setSelectedIndex( 0 );
2797 }
2798 }
```

Listing C.4: ClientInfo.java

```

1 package clientapp;
2
3 /**
4 *
5 * Creation date: (02-07-00 18:41:23)
6 * @author:
7 */
8 import java.util.Hashtable;
9 import java.util.Properties;
10 import java.io.File;
11 import java.util.HashSet;
12 import java.io.*;
13 import java.awt.datatransfer.*;
14
15 /**
16 * The main entity class for the client application. This class holds data structures and fields that
17 * must be shared by
18 * several client application components.
19 */
20 public class ClientInfo implements ClipboardOwner
21 {
22     private com.sun.media.jsdt.Session groupManagementSession = null;
23     private static java.util.Properties preferences = null;
24     private java.util.Hashtable groups = null;
25     private com.sun.media.jsdt.Channel groupManagementChannel = null;
26     private peerviewmisc.WorkGroup activeGroup = null;
```

```

26     private java.lang.String currentDocumentName = null;
27     private peerviewmisc.Message currentMessage = null;
28     private java.util.HashSet localFiles = new HashSet();
29     private java.awt.datatransfer.Clipboard clipBoard = new Clipboard( ClientConstants.getCLIPBOARD_NAME
30         () );
31     private boolean localMode = true;
32 /**
33 * ClientInfo constructor comment.
34 */
35 public ClientInfo() {
36     super();
37     initialize();
38 }
39 /**
40 * Create unique client id as concatenation of default name + system time in milliseconds + local host
41 * IP address or
42 * default name + system time if the IP address is unavailable.
43 * Creation date: (16-07-00 13:50:53)
44 * @return java.lang.String
45 */
46 public static String createUniqueClientID()
47 {
48     String ID = ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )+ (new java.util.Date
49         ().toString() );
50     try
51     {
52         ID = ID + java.net.InetAddress.getLocalHost().toString();
53     }
54     catch ( java.net.UnknownHostException UHE )
55     {
56         System.err.println( UHE.toString() );
57         System.exit( 0 );
58     }
59     return ID;
60 }
61 /**
62 * Insert the method's description here.
63 * Creation date: (16-07-00 13:38:00)
64 * @exception java.lang.Throwable The exception description.
65 */
66 protected void finalize() throws java.lang.Throwable
67 {
68     // write preferences
69 }
70 /**
71 * Insert the method's description here.
72 * Creation date: (06-07-00 13:15:33)
73 * @return peerviewmisc.WorkGroup
74 */
75 public peerviewmisc.WorkGroup getActiveGroup()
76 {
77     return activeGroup;
78 }
79 /**
80 * Insert the method's description here.
81 * Creation date: (07-08-00 15:57:43)
82 * @return java.awt.datatransfer.Clipboard
83 */
84 public java.awt.datatransfer.Clipboard getClipBoard()
85 {
86     return clipBoard;
87 }
88 /**
89 * Insert the method's description here.
90 * Creation date: (11-07-00 15:18:55)
91 * @return java.lang.String
92 */
93 public java.lang.String getCurrentDocumentName()

```

```

88     return currentDocumentName;
89 }
90 /**
91 * Insert the method's description here.
92 * Creation date: (19-07-00 17:30:43)
93 * @return peerviewmisc.Message
94 */
95 public peerviewmisc.Message getCurrentMessage() {
96     return currentMessage;
97 }
98 /**
99 * Insert the method's description here.
100 * Creation date: (04-07-00 17:42:59)
101 * @return com.sun.media.jsdt.Channel
102 */
103 public com.sun.media.jsdt.Channel getGroupManagementChannel() {
104     return groupManagementChannel;
105 }
106 /**
107 * Insert the method's description here.
108 * Creation date: (02-07-00 18:57:40)
109 * @return com.sun.media.jsdt.Session
110 */
111 public com.sun.media.jsdt.Session getGroupManagementSession() {
112     return groupManagementSession;
113 }
114 /**
115 * Insert the method's description here.
116 * Creation date: (04-07-00 16:21:40)
117 * @return java.util.Hashtable
118 */
119 public java.util.Hashtable getGroups() {
120     return groups;
121 }
122 /**
123 * Insert the method's description here.
124 * Creation date: (03-08-00 01:49:44)
125 * @return java.util.HashSet
126 */
127 public java.util.HashSet getLocalFiles() {
128     return localFiles;
129 }
130 /**
131 * Insert the method's description here.
132 * Creation date: (02-07-00 22:08:51)
133 * @return java.util.Properties
134 */
135 public static java.util.Properties getPreferences()
136 {
137     if ( preferences == null)
138     {
139         preferences = new Properties( ClientConstants.getDEFAULT_PREFERENCES() );
140         readPreferences();
141         // if the client has not yet been assigned a unique identifier, do so now. The new ID will be
142         // saved to disk upon exit
143         if ( (String) ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )).
144             equalsIgnoreCase( ClientConstants.getDEFAULT_CLIENT_NAME() )
145         {
146             ClientInfo.getPreferences().put( ClientConstants.getNAME(), createUniqueClientID() );
147             // System.out.println( ClientInfo.getPreferences().get( ClientConstants.getNAME() ) );
148         }
149     }
150     return preferences;
151 }
152 /**
153 * Insert the method's description here.

```

```

152 * Creation date: (16-07-00 13:36:57)
153 */
154 public void initialize()
155 {
156 }
157 /**
158 * Insert the method's description here.
159 * Creation date: (10-08-00 16:37:24)
160 * @return boolean
161 */
162 public boolean isLocalMode() {
163     return localMode;
164 }
165 /**
166 * Obligatory method for meeting the ClipboardOwner interface specification
167 * Creation date: (07-08-00 17:28:28)
168 * @param clipBoard java.awt.datatransfer.Clipboard
169 * @param contents java.awt.datatransfer.Transferable
170 */
171 public void lostOwnership(Clipboard clipBoard, Transferable contents)
172 {
173 }
174 /**
175 * 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
177 * member.
178 * This eliminates data redundancy and ensures consistency. The overhead is negligible as the file in
179 * question is essentially
180 * plain text.
181 * Algorithm:
182 * 1. Construct new Properties object P
183 * 2. Initialize P with default preferences
184 * 3. Read user preferences from disk file
185 * 4. IF read is successful THEN
186 *    4.1. Store user preferences in P
187 *    5. Return P
188 * This approach will return a valid object even if the read fails or in case of the application being
189 * used for the first
190 * time.
191 * Creation date: (25-06-00 13:54:55)
192 */
193 public static void readPreferences()
194 {
195     try
196     {
197         FileInputStream f = new FileInputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
198         preferences.load(f);
199         f.close();
200     }
201     catch (FileNotFoundException ffe)
202     {
203         // the exception can be handled by ignoring it since the default values in p will substitute for
204         // what should
205         // have been read from file.
206     }
207     catch (IOException ie)
208     {
209         // any IOException != FileNotFoundException should be dealt with in the conventional manner
210         System.err.println( ie.toString() );
211         System.exit(0);
212     }
213 }
214 /**
215 * Insert the method's description here.
216 * Creation date: (06-07-00 13:15:33)
217 * @param newActiveGroup peerviewmisc.WorkGroup

```

```

214  /*
215  public void setActiveGroup(peerviewmisc.WorkGroup newActiveGroup) {
216      activeGroup = newActiveGroup;
217 }
218 /**
219  * Insert the method's description here.
220  * Creation date: (07-08-00 15:57:43)
221  * @param newClipBoard java.awt.datatransfer.Clipboard
222  */
223 public void setClipBoard(java.awt.datatransfer.Clipboard newClipBoard) {
224     clipBoard = newClipBoard;
225 }
226 /**
227  * Insert the method's description here.
228  * Creation date: (11-07-00 15:18:55)
229  * @param newCurrentDocumentName java.lang.String
230  */
231 public void setCurrentDocumentName(java.lang.String newCurrentDocumentName) {
232     currentDocumentName = newCurrentDocumentName;
233 }
234 /**
235  * Insert the method's description here.
236  * Creation date: (19-07-00 17:30:43)
237  * @param newCurrentMessage peerviewmisc.Message
238  */
239 public void setCurrentMessage(peerviewmisc.Message newCurrentMessage) {
240     currentMessage = newCurrentMessage;
241 }
242 /**
243  * Insert the method's description here.
244  * Creation date: (04-07-00 17:42:59)
245  * @param newGroupManagementChannel com.sun.media.jsdt.Channel
246  */
247 public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
248     groupManagementChannel = newGroupManagementChannel;
249 }
250 /**
251  * Insert the method's description here.
252  * Creation date: (02-07-00 18:57:40)
253  * @param newGroupManagementSession com.sun.media.jsdt.Session
254  */
255 public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
256     groupManagementSession = newGroupManagementSession;
257 }
258 /**
259  * Insert the method's description here.
260  * Creation date: (04-07-00 16:21:40)
261  * @param newGroups java.util.Hashtable
262  */
263 public void setGroups(java.util.Hashtable newGroups) {
264     groups = newGroups;
265 }
266 /**
267  * Insert the method's description here.
268  * Creation date: (03-08-00 01:49:44)
269  * @param newLocalFiles java.util.HashSet
270  */
271 public void setLocalFiles(java.util.HashSet newLocalFiles) {
272     localFiles = newLocalFiles;
273 }
274 /**
275  * Insert the method's description here.
276  * Creation date: (10-08-00 16:37:24)
277  * @param newLocalMode boolean
278  */
279 public void setLocalMode(boolean newLocalMode) {

```

```

280     localMode = newLocalMode;
281 }
282 /**
283 * Insert the method's description here.
284 * Creation date: (02-07-00 22:08:51)
285 * @param newPreferences java.util.Properties
286 */
287 public static void setPreferences(java.util.Properties newPreferences) {
288     preferences = newPreferences;
289 }
290 /**
291 * Write preferences to disk file.
292 * Creation date: (17-07-00 12:13:32)
293 * @param preferences java.util.Properties
294 */
295 public static void writePreferences()
296 {
297     // write Properties object to disk
298     try
299     {
300         FileOutputStream f = new FileOutputStream( ClientConstants.getPREFERENCES_FILE_NAME() );
301         preferences.store( f, ClientConstants.getPropertyHeader() );
302         f.close();
303     }
304     catch (IOException ie)
305     {
306         System.err.println( ie.toString() );
307         System.exit(0);
308     }
309 }
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 /**
25 * The main control class for the client application. This class holds methods and control structures
26 * shared by several
27 * client application components.
28 * Creation date: (14-06-00 21:27:43)
29 * @author:
30 */
31 public class ClientManager
{

```

```

32  /* public class DocPackage implements Serializable
33  {
34      String docAndPathName;
35      java.awt.Rectangle bounds;
36
37      public DocPackage( String docAndPathNameArg, java.awt.Rectangle boundsArg )
38      {
39          docAndPathName = docAndPathNameArg;
40          bounds = boundsArg;
41      }
42
43      public String getDocAndPathName()
44      {
45          return docAndPathName;
46      }
47
48      public java.awt.Rectangle getBounds()
49      {
50          return bounds;
51      }
52
53      public void setDocAndPathName( String docAndPathNameArg )
54      {
55          docAndPathName = docAndPathNameArg;
56      }
57
58      public void setBounds( java.awt.Rectangle boundsArg )
59      {
60          bounds = boundsArg;
61      }
62  };
63 */
64
65     /** Event handler for JSDT connections. Ensures automatic notification in case of connection failure
66     */
67     public class GroupManagementListener implements ConnectionListener
68     {
69         public void connectionFailed( ConnectionEvent ce )
70         {
71             groupManagementConnectionFailed( ce );
72         }
73     };
74
75     // the below inner classes derived from Thread are used instead of member methods to allow execution
76     // in separate threads.
77     // This is necessary to allow the event-dispatching thread to process events and thus enable proper
78     // updating of the message area progress bar
79     // in the client main window. Initially, I used paintImmediately to achieve the desired effect but
80     // that seemed to produce flicker on some platforms and
81     // so had to be abandoned.
82     /** An object of this class should be instantiated with a set of documents and launched in a separate
83     // thread so that
84     * it executes without blocking the event dispatching thread and thus the GUI.
85     */
86     public class RemoveDistributedDocumentsFromPanorama extends Thread
87     {
88         public void run()
89         {
90             owner.lockInterface( true );
91             java.util.Set keys = owner.getPanoramaPanel().getDocuments().keySet();
92             Iterator iterator = keys.iterator();
93             owner.initMessageArea( 1, owner.getPanoramaPanel().getDocuments().size(), 1, ClientConstants.
94                 getREMOVING_DISTRIBUTED_DOCUMENTS_MESSAGE() );
95             // copy to array first to avoid concurrentModificationException
96             // this should be considered an interim solution till I come up with a cleaner alternative
97             String keyArray[] = new String[ keys.size() ];

```

```

94         for ( int i = 0 ; iterator.hasNext(); i++ )
95         {
96             keyArray[i] = (String) iterator.next();
97         }
98         for ( int i = 0; i < keyArray.length; i++ )
99         {
100             String key = keyArray[i];
101             edu.umd.cs.jazz.ZVisualLeaf leaf = (edu.umd.cs.jazz.ZVisualLeaf) owner.getPanoramaPanel().
102                 getDocuments().get( key );
103             owner.getPanoramaPanel().removeGlobalDocument( (String) leaf.getClientProperty(
104                 ClientConstants.getSENDER_NAME() ),
105                                         (String) leaf.getClientProperty( ClientConstants.
106                                             getDOCUMENT_NAME() ) );
107             owner.advanceTask();
108             yield();
109         }
110     }
111     /** An object of this class should be instantiated with a set S of file names and launched in a
112      * separate thread. It will
113      * add the files whose names are in S to the panorama.
114     */
115     public class AddDocuments extends Thread
116     {
117         File newFiles[];
118
119         public AddDocuments( File newFilesArg[] )
120         {
121             newFiles = newFilesArg;
122         }
123
124         public void run()
125         {
126             owner.lockInterface( true );
127             boolean duplicateFlag = false;
128             // prepare progress indicator for the document retrieval process
129             owner.initMessageArea( 0, newFiles.length, 0, ClientConstants.getRETRIEVING_DOCUMENTS() );
130             java.util.HashSet fileSet = new java.util.HashSet();
131             for (int i = 0; i < newFiles.length; i++)
132             {
133                 owner.advanceTask();
134                 if ( getClientInfo().getLocalFiles().contains( newFiles[i] )== true )
135                 {
136                     // if file is already in the panorama, raise the flag and continue with the next file in
137                     // the array
138                     duplicateFlag = true;
139                     continue;
140                 }
141                 else
142                 {
143                     getClientInfo().getLocalFiles().add( newFiles[i] );
144                 }
145                 try
146                 {
147                     owner.getPanoramaPanel().addLocalDocument( readDocument( newFiles[i] ),
148                                         ClientInfo.getPreferences().getProperty( ClientConstants.
149                                             getName() ),
150                                         ClientInfo.getPreferences().getProperty( ClientConstants.
151                                             getAUTHOR_NAME() ),
152                                         newFiles[i].getPath(),
153                                         newFiles[i].lastModified()
154                                     );
155                 }
156             }

```

```

152         catch ( IOException IOE )
153         {
154             }
155             informServerOfNewDocument( newFiles[i].getPath() );
156             fileSet.add( newFiles[i] );
157             yield();
158         }
159         broadcastDocuments( fileSet );
160         updatePanorama();
161         // System.out.println("inde i addDocuments åtrd");
162         owner.resetMessageArea();
163         if ( duplicateFlag == true )
164         {
165             MessageBox messageBox = new MessageBox();
166             messageBox.setText( ClientConstants.getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() );
167             messageBox.show();
168         }
169         owner.lockInterface( false );
170     }
171 }
172 /**
173 * This class should be instantiated with a set S of files and launched in a separate thread. It
174 * will remove
175 * from the panorama the documents corresponding the members of S.
176 */
177 public class RemoveDocuments extends Thread
178 {
179     File[] removables;
180
181     public RemoveDocuments( File[] removablesArg )
182     {
183         removables = removablesArg;
184     }
185
186     public void run()
187     {
188         owner.lockInterface( true );
189         owner.getPanoramaPanel().setPanoramaLock( true );
190         owner.initMessageArea( 1, removables.length, 1, ClientConstants.getREMOVING_DOCUMENTS() );
191         java.util.Vector filePaths = new java.util.Vector();
192         for ( int i = 0 ; i < removables.length; i++ )
193         {
194             owner.getPanoramaPanel().removeLocalDocument( ClientInfo.getPreferences().getProperty(
195                 ClientConstants.getNAME() ), removables[i].getPath() );
196             getClientInfo().getLocalFiles().remove( removables[i] );
197             filePaths.add( removables[i].getPath() );
198             owner.advanceTask();
199             yield();
200         }
201         if ( getClientInfo().isLocalMode() == false )
202         {
203             broadcastDocumentRemovals( filePaths );
204         }
205         updatePanorama();
206         owner.resetMessageArea();
207         owner.getPanoramaPanel().setPanoramaLock( false );
208         owner.lockInterface( false );
209     }
210 }
211 /**
212 * This class should be instantiated with a recipient ID and author name and launched in a separate
213 * thread. It
214 * will send to the recipient the set of local documents currently displayed in the panorama.
215 */
216 public class UnicastDocuments extends Thread
217 {
218     String recipientName;
219     String recipientAuthorName;

```

```

216     public UnicastDocuments( String recipientNameArg, String recipientAuthorNameArg )
217     {
218         recipientName = recipientNameArg;
219         recipientAuthorName = recipientAuthorNameArg;
220     }
221
222     public void run()
223     {
224         if ( getClientInfo().getLocalFiles().size() > 0 )
225         {
226             WorkGroup activeGroup = getClientInfo().getActiveGroup();
227             urlString url = null;
228             Session session = null;
229             Channel groupChannel = null;
230             java.util.Vector rawDocumentData = new java.util.Vector();
231             try
232             {
233                 url = activeGroup.getSessionURL();
234                 session = SessionFactory.createSession( getGroupManagementClient(), url, false );
235                 groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
236                     getWorkGroupChannelID(), true, true, false );
237             }
238             catch ( JSDTEException JSDTE )
239             {
240                 handleJSDTEException( JSDTE );
241                 return;
242             }
243             owner.initMessageArea(1, getClientInfo().getLocalFiles().size(), 1,
244                     ClientConstants.getUNICASTING_DOCUMENTS_MESSAGE( getClientInfo().
245                         getLocalFiles().size(),
246                         recipientAuthorName ) );
247             Iterator iterator = getClientInfo().getLocalFiles().iterator();
248             while ( iterator.hasNext() )
249             {
250                 File file = (File)iterator.next();
251                 // read contents of file and compress it
252                 try
253                 {
254                     byte fileContents[] = readDocument( file );
255                     CompressedDocumentPackage compressedDocumentPackage = deflateDocument( fileContents );
256                     compressedDocumentPackage.setPath( file.getPath() );
257                     // separate name from path here to avoid confusion over separator char across platforms
258                     .
259                     compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf(
260                         ( File.separator )+ 1 ) );
261                     rawDocumentData.add( compressedDocumentPackage );
262                     owner.advanceTask();
263                     yield();
264                 }
265                 catch ( IOException IOE )
266                 {
267                     System.err.println( IOE.toString() );
268                     System.exit( 0 );
269                 }
270             }
271             try
272             {
273                 DataPackage outgoingPackage = new DataPackage();
274                 outgoingPackage.setContents( rawDocumentData );
275                 outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants
276                     .getNEW_DOCUMENTS() );
277                 outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), getClientInfo()
278                     .getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ) );
279                 Data data = new Data( outgoingPackage );
280                 groupChannel.sendToClient( getGroupManagementClient(), recipientName, data );
281             }

```

```

        }
    catch ( JSDTEException JSDTE )
    {
        handleJSDTEException( JSDTE );
    }
    owner.resetMessageArea();
}
};

/** Use this class to remove non-local documents from the panorama.
 * 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
 * launched in a separate thread.
 */
290 public class RemoveDistributedDocuments extends Thread
{
    String authorName;
    String senderName;
    java.util.Vector filePaths;

296    public RemoveDistributedDocuments(String authorNameArg, String senderNameArg, java.util.Vector
        filePathsArg )
    {
        authorName = authorNameArg;
        senderName = senderNameArg;
300        filePaths = filePathsArg;
    }

302    public void run()
    {
        owner.lockInterface( true );
        owner.initMessageArea( 1, filePaths.size(), 1, ClientConstants.
            getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST( authorName, filePaths.size() ) );
        Iterator iterator = filePaths.iterator();
308        while ( iterator.hasNext() )
        {
            owner.getPanoramaPanel().removeGlobalDocument( senderName, (String) iterator.next() );
            owner.advanceTask();
            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
    {
        DataPackage incomingPackage = null;
        String senderName = null;
326        public AddDistributedDocuments( DataPackage incomingPackageArg, String senderNameArg )
        {
            incomingPackage = incomingPackageArg;
            senderName = senderNameArg;
        }

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

```

```

336     java.util.Vector documents = (java.util.Vector) incomingPackage.getContents();
337     String authorName = incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() );
338     owner.initMessageArea( 1, documents.size(), 1, ClientConstants.getADDING_DISTRIBUTED_DOCUMENTS(
339         authorName, documents.size() ) );
340     for ( int i = 0 ; i < documents.size(); i++ )
341     {
342         CompressedDocumentPackage compressedDocumentPackage = (CompressedDocumentPackage) documents.
343             get( i );
344         byte[] rawDocumentData = inflateDocument( compressedDocumentPackage );
345         owner.getPanoramaPanel().addGlobalDocument( rawDocumentData, senderName, authorName,
346             compressedDocumentPackage.getPath(), compressedDocumentPackage.getName() );
347         owner.advanceTask();
348         yield();
349     }
350 };
351
352 /** Client object to be used with the group management session.
353 */
354 public class GroupManagementClient implements Client, ChannelConsumer
355 {
356     String name;
357
358     public GroupManagementClient()
359     {
360     }
361
362     public String getName()
363     {
364         return name;
365     }
366
367     public void setName( String nameArg )
368     {
369         name = nameArg;
370     }
371
372     /** Accept and parse incoming data. The if-else structure maps message types to functions.
373 */
374     public synchronized void dataReceived( Data data )
375     {
376         DataPackage incomingPackage = null;
377         try
378         {
379             incomingPackage = (DataPackage) data.getDataAsObject();
380         }
381         catch ( Exception E )
382         {
383             System.err.println( E.toString() );
384             System.exit(0);
385         }
386         if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ) .
387             equalsIgnoreCase ( ClientConstants.getGROUP_DIRECTORY() ) )
388         {
389             updateGroupDirectory( (Hashtable) incomingPackage.getContents() );
390         }
391         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ) .
392             equalsIgnoreCase ( ClientConstants.getNEW_DOCUMENT() ) )
393         {
394             addDistributedDocument( incomingPackage, data.getSenderName() );
395         }
396         else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ) .
397             equalsIgnoreCase ( ClientConstants.getNEW_DOCUMENTS() ) )

```

```

    {
      (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.getDOC PATH() ) );
  }
 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 {
 415   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 {
 419   (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 {
 423   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 {
 427   updateDistributedDocument( incomingPackage.getProperties().getProperty( DataPackage .
    getDOC PATH() ), data.getSenderName(), (CompressedDocumentPackage) incomingPackage .
    getContents() );
  }
 428
  else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ) .
    equalsIgnoreCase( ClientConstants.getTOPIC_TREE_UPDATE() ))
 430 {
 431   updateTopicTree( data.getSenderName(), (String) incomingPackage.getProperties().getProperty(
    DataPackage.getDOC PATH() ), (DefaultTreeModel) incomingPackage.getContents() );
  }
 432
  else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ) .
    equalsIgnoreCase( ClientConstants.getUPDATE_PANORAMA() ))
 434 {
 435   updatePanoramaByRequest();
  }
 436
  }
 438
  public Object authenticate ( AuthenticationInfo ai )
 440 {
 441   return null;
  }

```

```

    };

444     private ClientApplication owner;
445     private ClientInfo clientInfo = null;
446     private GroupManagementClient groupManagementClient = null;
447 /**
448 * ClientManager constructor comment.
449 */
450 public ClientManager(ClientApplication newOwner)
451 {
452     super();
453     owner = newOwner;
454     initialize();
455 }
456 /**
457 * Inflate a data package received over a network connection and add it to the panorama using the data
458 * package properties
459 * as descriptive parameters.
460 * Creation date: (08-07-00 18:08:33)
461 * @param contents byte[]
462 * @param path java.lang.String[]
463 */
464 public void addDistributedDocument(DataPackage incomingPackage, String sender)
465 {
466     byte[] rawFileContents = inflateDocument( (CompressedDocumentPackage) incomingPackage.getContents() )
467             ;
468     owner.getPanoramaPanel().addGlobalDocument( rawFileContents,
469             sender,
470             incomingPackage.getProperties().getProperty( DataPackage.getAUTHOR_NAME() ),
471             incomingPackage.getProperties().getProperty( DataPackage.getDOC_PATH() ),
472             incomingPackage.getProperties().getProperty( DataPackage.getDOCNAME() ) );
473     owner.initMessageArea( 1, 1, 1, ClientConstants.getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(
474         incomingPackage.getProperties().getProperty( DataPackage.getDOC_PATH() ), incomingPackage.
475             getProperties().getProperty( DataPackage.getAUTHOR_NAME() ) ) );
476 }
477 /**
478 * Add documents to the panorama by launching a separate thread to handle the task.
479 * Creation date: (03-10-00 11:17:31)
480 * @param newFiles java.io.File[]
481 */
482 public void addDocuments(File[] newFiles)
483 {
484     AddDocuments addDocuments = new AddDocuments( newFiles );
485     addDocuments.start();
486 /* try
487 {
488     addDocuments.join();
489 }
490 catch ( InterruptedException ie )
491 {
492     System.err.println( ie.toString() );
493 }
494 */
495 /**
496 * Read a listing of document names from disk and add the corresponding documents to the panorama.
497 * Creation date: (11-10-00 13:19:00)
498 * @param docs java.util.Vector
499 */
500 public void addDocumentsFromDisk(java.util.Vector docs)
501 {
502     if ( docs != null )
503     {
504         File files[] = new File[ docs.size() ];
505         for ( int i = 0 ; i < docs.size(); i++ )
506         {

```

```

506         DocPackage docPackage = (DocPackage) docs.get( i );
507         File file = new File( docPackage.getDocNameAndPath() );
508         files[ i ] = file;
509     }
510     AddDocuments addDocuments = new AddDocuments( files );
511     addDocuments.start();
512     try
513     {
514         addDocuments.join();
515     }
516     catch ( InterruptedException ie )
517     {
518         System.err.println( ie.toString() );
519     }
520     // the below assumes that the localdocuments set now contains all the newly added documents from
521     // the above call to
522     // addDocuments. A cleaner solution would be to merge the adding of files with the repositioning
523     // but the
524     // readDocumentList feature wasn't included in the original design, so pending a revision of the
525     // code, this should do OK.
526     for ( int i = 0 ; i < docs.size(); i++ )
527     {
528         DocPackage docPackage = (DocPackage) docs.get( i );
529         String fqName = ClientConstants.createFullyQualifiedDocumentName( ClientInfo.getPreferences().
530             getProperty( ClientConstants.getNAME() ),
531             docPackage.getDocNameAndPath() );
532         edu.umd.cs.jazz.ZVisualLeaf leafToBeRepositioned = (edu.umd.cs.jazz.ZVisualLeaf) owner.
533             getPanoramaPanel().getLocalDocuments().get( fqName );
534         ( (PZSwing) leafToBeRepositioned.getVisualComponent() ).getComponent().setSize( docPackage.
535             getSize() );
536         leafToBeRepositioned.editor().getTransformGroup().setTranslation( (float) docPackage.
537             getPosition().getWidth(),
538             (float) docPackage.getPosition().getHeight() );
539         leafToBeRepositioned.putClientProperty( ClientConstants.getNodeLocked(), docPackage.
540             getNodeLocked() );
541     }
542     owner.getPanoramaPanel().validate();
543     owner.getPanoramaPanel().repaint();
544 }
545 /**
546  * Send of list of files to be removed from the panorama to all members of the group this client is
547  * joined to, if any.
548  * Creation date: (26-07-00 02:30:02)
549  * @param files java.io.File[]
550  */
551 public void broadcastDocumentRemovals(java.util.Vector files)
552 {
553     Session session = null;
554     Channel groupChannel = null;
555     if ( ( getClientInfo().isLocalMode() == false )&&
556         ( files.size() > 0 ) )
557     {
558         WorkGroup activeGroup = getClientInfo().getActiveGroup();
559         owner.initMessageArea(1, 1, 1, ClientConstants.getBROADCASTING_DOCUMENT_REMOVALS() );
560         DataPackage outgoingPackage = new DataPackage();
561         outgoingPackage.setContents( files );
562         outgoingPackage.getProperties().setProperty( DataPackage.getDescription(), ClientConstants.
563             getREMOVE_DOCUMENTS() );
564         outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), activeGroup.
565             getWorkGroupID() );
566         outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(),
567             ClientInfo.getPreferences().getProperty( ClientConstants.
568                 getAUTHOR_NAME() ) );
569         outgoingPackage.getProperties().setProperty( DataPackage.getClientName(),
570             ClientInfo.getPreferences().getProperty( ClientConstants.
571                 getClientName() ) );
572     }

```

```

560             ClientInfo.getPreferences().getProperty( ClientConstants.getNAME() )
561             );
562     try
563     {
564         urlString = getClientInfo().getActiveGroup().getSessionURL();
565         session = SessionFactory.createSession( getGroupManagementClient(), url, false );
566         groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
567             getWorkGroupChannelID(), true, true, false);
568         Data data = new Data( outgoingPackage );
569         groupChannel.sendToOthers( getGroupManagementClient(), data );
570     }
571     catch ( JSDTEException JSDTE )
572     {
573         handleJSDTEException( JSDTE );
574     }
575     owner.resetMessageArea();
576 }
577 /**
578 * Transmit a set of files to all members of the group that this client belongs to, if any.
579 * Creation date: (06-07-00 14:06:11)
580 */
581 public void broadcastDocuments( java.util.HashSet files)
582 {
583     if ( ( getClientInfo().isLocalMode() == false )&& ( files.size() > 0 ) )
584     {
585         WorkGroup activeGroup = getClientInfo().getActiveGroup();
586         urlString = null;
587         Session session = null;
588         Channel groupChannel = null;
589         String clientNames[] = null;
590         int numClientNames = 0;
591         java.util.Vector rawDocumentData = new java.util.Vector();
592         try
593         {
594             url = getClientInfo().getActiveGroup().getSessionURL();
595             session = SessionFactory.createSession( getGroupManagementClient(), url, false );
596             groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
597                 getWorkGroupChannelID(), true, true, false);
598         }
599         catch ( JSDTEException JSDTE )
600         {
601             handleJSDTEException( JSDTE );
602         }
603         try
604         {
605             numClientNames = groupChannel.listClientNames().length;
606         }
607         catch ( JSDTEException JSDTE )
608         {
609             handleJSDTEException( JSDTE );
610         }
611         // inform user that broadcast is in progress
612         if ( numClientNames > 1 )
613         {
614             owner.initMessageArea(1, files.size(), 1, ClientConstants.getBROADCASTING_DOCUMENTS_MESSAGE(
615                 files.size() ));
616         }
617         Iterator iterator = files.iterator();
618         while ( iterator.hasNext() )
619         {
620             File file = (File) iterator.next();
621             // if more members than this client
622             CompressedDocumentPackage compressedDocumentPackage = null;
623             try

```

```

622     {
623         // read contents of file and compress it
624         byte fileContents[] = readDocument( file );
625         compressedDocumentPackage = deflateDocument( fileContents );
626         compressedDocumentPackage.setPath( file.getPath() );
627         // separate name from path here to avoid confusion over separator char across platforms.
628         compressedDocumentPackage.setName( file.getPath().substring( file.getPath().lastIndexOf( File.
629             separator )+ 1 ) );
630         //System.out.println( "compressedPackageName" + compressedDocumentPackage.getName() );
631     }
632     catch ( IOException IOE )
633     {
634         System.err.println( IOE.toString() );
635         System.exit( 0 );
636     }
637     rawDocumentData.add( compressedDocumentPackage );
638     if ( numClientNames > 1 )
639     {
640         owner.advanceTask();
641     }
642     // submit raw document data
643     if ( numClientNames > 1 )
644     {
645         DataPackage outgoingPackage = new DataPackage();
646         outgoingPackage.setContents( rawDocumentData );
647         outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
648             getNEW_DOCUMENTS() );
649         outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(),
650                         ClientInfo.getPreferences().getProperty( ClientConstants.
651                             getAUTHOR_NAME() ) );
652         outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
653             getActiveGroup().getWorkGroupID() );
654         try
655         {
656             Data data = new Data( outgoingPackage );
657             // System.out.println( "Broadcasted" );
658             groupChannel.sendToOthers( getGroupManagementClient(), data );
659         }
660         catch ( JSDTEException JSDTE )
661         {
662             handleJSDTEException( JSDTE );
663         }
664     }
665 }
666 /**
667 * Send a difference array to the other members of the group that the client belongs to, if any, so that
668 * they can
669 * update their copy of the relevant document. This method is called when a document has been modified
670 * to keep the
671 * remote copies in synch with the master at the author client.
672 * Creation date: (21-08-00 16:18:43)
673 * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
674 */
675 public void broadcastDocumentUpdate( String documentName, CompressedDocumentPackage
676     compressedDifferenceBitmapPackage )
677 {
678     DataPackage outgoingPackage = new DataPackage();
679     outgoingPackage.setContents( compressedDifferenceBitmapPackage );

```

```

        outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
            getDOCUMENT_UPDATE() );
680    outgoingPackage.getProperties().setProperty( DataPackage.getDOCSPATH(), 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 ( JSDTEException JSDTE )
692    {
        handleJSDTEException( JSDTE );
694    }

696 }
/***
698 * Set all properties of the topic tree to null so that it appears empty to the user.
699 * Creation date: (01-08-00 15:53:33)
700 */
701 public void clearTopicTree()
702 {
    owner.setMessage( null );
704    getClientInfo().setCurrentMessage( null );
    getClientInfo().setCurrentDocumentName( null );
706    owner.setTopicTree( null );
}
708 /**
709 * 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
711 * array.
712 * Then appends the remainder of newRawFileContents, if any, to the resulting bitmap and returns it.
713 * This method is meant for use in the document update phase, both in the transmitting and receiving end
714 * Specifically, the sender computes the difference between an old and a new version of a document and
715 * broadcasts it
716 * in gzipped form to one or more recipients who then apply the below method to their old version and
717 * the difference
718 * bitmap, ending up with a transformed document identical to the sender's version.
719 * Creation date: (21-08-00 16:18:01)
720 * @return byte[]
721 * @param oldRawFileContents byte[]
722 * @param newRawFileContents byte[]
723 */
724 public byte[] computeBitmapCombination(byte[] oldRawFileContents, byte[] newRawFileContents)
725 {
    int shortestArrayLength;
726    int longestArrayLength;
    if ( oldRawFileContents.length < newRawFileContents.length )
727    {
        shortestArrayLength = oldRawFileContents.length;
728        longestArrayLength = newRawFileContents.length;
    }
729    else
730    {
        shortestArrayLength = newRawFileContents.length;
        longestArrayLength = oldRawFileContents.length;
    }
731    byte differenceBitmap[] = new byte[ newRawFileContents.length ];
732    // XOR the two file contents arrays and store the result in the differenceBitmap array
733    int i;
734    for ( i = 0 ; i < shortestArrayLength; i++ )

```

```

    {
740     byte oldByte = oldRawFileContents[i];
741     byte newByte = newRawFileContents[i];
742     // ( a AND b )OR ( !a AND !b )== XOR( a,b )
743     differenceBitmap[i] = (byte) (
744         ( (byte) ( oldByte | newByte ))&
745         ( (byte) ( (Byte.MAX_VALUE - oldByte) | (Byte.MAX_VALUE - newByte ))));
746     }
747     // copy remainder of new file contents, if any, to differenceBitmap array
748     for ( ; i < newRawFileContents.length; i++ )
749     {
750         differenceBitmap[i] = newRawFileContents[i];
751     }
752     return differenceBitmap;
753 }
754 /**
755 * Perform various initialization maneuvers at start-up
756 * Creation date: (21-10-00 21:35:37)
757 */
758 public void configure()
759 {
760     connectToGroupManagement();
761     requestGroupsFromServer();
762     connectToGroup();
763     addDocumentsFromDisk( readDocumentList() );
764 }
765 /**
766 * Connect to the most recently joined group, if any, as specified in the preferences.
767 * Creation date: (09-08-00 23:48:29)
768 */
769 public void connectToGroup()
770 {
771     if ( getClientInfo().getGroups() != null )
772     {
773         String activeGroupUponExitID = (String) ClientInfo.getPreferences().getProperty( ClientConstants.
774             getACTIVE_GROUP_UPON_EXIT() );
775         WorkGroup workGroup = (WorkGroup) getClientInfo().getGroups().get( activeGroupUponExitID );
776         if ( activeGroupUponExitID.equalsIgnoreCase( ClientConstants.getDEFAULT_GROUP_ID() )== false )
777         {
778             connectToGroup( workGroup );
779         }
780         else
781         {
782             getClientInfo().setActiveGroup( null );
783             informUserOfLocalMode();
784         }
785     }
786 }
787 /**
788 * Connect to the workgroup specified by the parameter, if possible.
789 * Creation date: (10-08-00 00:35:15)
790 * @param groupURL com.sun.media.jsdt.URLString
791 */
792 public void connectToGroup( WorkGroup workGroup )
793 {
794     Session session;
795     Channel channel;
796     try
797     {
798         urlString url = workGroup.getSessionURL();
799         session = SessionFactory.createSession( getGroupManagementClient(), url, true );
800         channel = session.createChannel( getGroupManagementClient(),
801             workGroup.getWorkGroupChannelID() ,
802             true, true, true);
803         channel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
804         getClientInfo().setActiveGroup( workGroup );
805     }

```

```

804     getClientInfo().setLocalMode( false );
805     // broadcast to other group members
806     broadcastDocuments( getClientInfo().getLocalFiles() );
807     // request documents from other members of group
808     requestDocuments();
809     // announce successful join
810     informUserOfGroupJoin( workGroup );
811 }
812 catch ( JSDTEException JSDTE )
813 {
814     handleJSDTEException( JSDTE );
815 }
816
817 }
818 /**
819 * Connect the client to the server via the group management session.
820 * Creation date: (02-07-00 19:16:42)
821 */
822 public void connectToGroupManagement()
823 {
824     Session session = null;
825     Channel groupManagementChannel = null;
826     if ( getClientInfo().getGroupManagementSession() == null )
827     {
828         try
829         {
830             /* if disconnect first if already connected
831             if ( getClientInfo().getGroupManagementSession() != null )
832             {
833                 if ( getClientInfo().isLocalMode() == false )
834                 {
835                     disconnectFromGroup( getClientInfo().getActiveGroup() );
836                 }
837                 clientInfo.getGroupManagementSession().leave( getGroupManagementClient() );
838                 /*clientInfo.getGroupManagementChannel().leave( getGroupManagementClient() );
839                 clientInfo.setGroupManagementSession( null );
840                 clientInfo.setGroupManagementChannel( null );
841             }
842         */
843         urlString = urlString.createSessionURL( (String)ClientInfo.getPreferences().getProperty(
844             ClientConstants.getServer_NAME() ),
845                         Integer.parseInt( (String) (ClientInfo.getPreferences().
846                             getProperty( ClientConstants.getServer_PORT() ))),
847                         (String)ClientInfo.getPreferences().getProperty( ClientConstants.
848                             getConnection_TYPE() ),
849                         ClientConstants.getGROUP_MANAGEMENT_SESSION_NAME() );
850         if ( SessionFactory.sessionExists( urlString ) )
851         {
852             session = SessionFactory.createSession( getGroupManagementClient(), urlString, true );
853             groupManagementChannel = session.createChannel( getGroupManagementClient(), ClientConstants.
854                 getGROUP_CHANNEL_NAME(), true, true, true );
855             groupManagementChannel.addConsumer( getGroupManagementClient(), getGroupManagementClient() );
856             owner.initMessageArea( 1, 1, 1, ClientConstants.getCONNECTED_TO_GROUP_MANAGEMENT() );
857         }
858         else
859         {
860             informUserOfLocalMode();
861             owner.initMessageArea( 1, 1, 1, ClientConstants.getFAILED_TO_CONNECT_TO_SERVER() );
862         }
863     }
864     catch ( NameInUseException NIUE )
865     {
866         owner.initMessageArea( 1, 1, 1, ClientConstants.getNOTIFYING_SERVER_OF_NAME_IN_USE() );
867         informServerOfExit();
868     }

```

```

866     catch ( JSDTEException JSDTE )
867     {
868         handleJSDTEException( JSDTE );
869         informUserOfLocalMode();
870     }
871
872     getClientInfo().setGroupManagementSession(session);
873     getClientInfo().setGroupManagementChannel(groupManagementChannel);
874 }
875
876 /**
877 * Copy the text selected in the visual component associated with the selectedNode parameter to the
878 * client
879 * clipboard.
880 * Creation date: (07-08-00 17:01:17)
881 * @param selectedNode edu.umd.cs.jazz.ZVisualLeaf
882 */
883 public void copyText(edu.umd.cs.jazz.ZVisualLeaf selectedNode)
884 {
885     JScrollPane selectedPane = (JScrollPane) ( (PZSwing) selectedNode.getVisualComponent() ).getComponent
886     ();
887     // if the selected node is capable of containing text. The check is mostly to be on the safe side; it
888     // is assumed that
889     // the caller tries to ensure that the copy text menu item is invoked only on text documents.
890     if ( selectedPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
891     {
892         String selectedText = ( (javax.swing.text.JTextComponent) selectedPane.getViewport().getView() ).
893             getSelectedText();
894         if ( selectedText != null )
895         {
896             ( (javax.swing.text.JTextComponent) selectedPane.getViewport().getView() ).copy();
897             owner.initMessageArea( 1, 1, 1, ClientConstants.getCOPIED_TEXT_MESSAGE() );
898         }
899         else
900         {
901             owner.initMessageArea( 1, 1, 1, ClientConstants.getNO_TEXT_TO_COPY_MESSAGE() );
902         }
903     }
904 /**
905 * Submit a request to the server that a group be created and added to the group directory.
906 * Creation date: (17-07-00 17:25:50)
907 * @param workGroup peerviewmisc.WorkGroup
908 */
909 public void createWorkGroup(WorkGroup workGroup)
910 {
911     DataPackage outgoingPackage = new DataPackage();
912     outgoingPackage.setContents( workGroup );
913     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
914         getRequest_ADD_GROUP() );
915     try
916     {
917         if ( getClientInfo().getGroupManagementChannel() != null )
918         {
919             getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
920                 ClientInfo.getPreferences().getProperty( ClientConstants.
921                     getSERVER_NAME() ),
922                 new Data( outgoingPackage ) );
923         }
924     }
925     catch ( JSDTEException JSDTE )
926     {
927         handleJSDTEException( JSDTE );
928     }

```

```

926 }
927 /**
928 * Compress and package the contents of an array representing a document.
929 * Creation date: (08-07-00 20:49:16)
930 * @return CompressedDocumentPackage
931 * @param uncompressedContents byte
932 */
933 public CompressedDocumentPackage deflateDocument(byte[] uncompressedContents)
934 {
935     Deflater deflater = new Deflater( Integer.parseInt( ClientInfo.getPreferences().getProperty(
936         ClientConstants.getCOMPRESSION_LEVEL() ) ) );
937     byte compressedContents[] = new byte[ uncompressedContents.length ];
938
939     deflater.setInput( uncompressedContents, 0 , uncompressedContents.length );
940     deflater.finish();
941     int compressedSize = deflater.deflate( compressedContents );
942     byte compressedAndCondensedContents[] = new byte[ compressedContents.length ];
943     for ( int i = 0 ; i < compressedSize; i++ )
944     {
945         compressedAndCondensedContents[ i ] = compressedContents[ i ];
946     }
947     CompressedDocumentPackage compressedDocumentPackage = new CompressedDocumentPackage();
948
949     compressedDocumentPackage.setCompressedContents( compressedAndCondensedContents );
950     compressedDocumentPackage.setCompressedLength( compressedSize );
951     compressedDocumentPackage.setUncompressedLength( uncompressedContents.length );
952
953     // System.out.println( "Original: " + uncompressedContents.length + " Deflated: " + compressedSize );
954     return compressedDocumentPackage;
955 }
956 /**
957 * Leave a work group (passed as argument).
958 * Creation date: (10-08-00 21:14:06)
959 * @param workGroup peerviewmisc.WorkGroup
960 */
961 public void disconnectFromGroup(WorkGroup workGroup)
962 {
963     if ( workGroup != null )
964     {
965         // broadcastDocumentRemovals( getClientInfo().getLocalFiles() );
966         // the next step is carried out to allow the broadcasts to conclude - they are apparently
967         // disrupted if the leave is executed prematurely
968         removeDistributedDocumentsFromPanorama();
969         // obtain reference to the session that this client currently subscribes to and leave it. This
970         // will automatically
971         // expel the client from any objects subordinate to that session.
972         try
973         {
974             urlString = workGroup.getSessionURL();
975             Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
976             session.leave( getGroupManagementClient() );
977         }
978         catch ( JSDTEexception JSDTE )
979         {
980             System.err.println( JSDTE.toString() );
981             System.exit( 0 );
982         }
983     }
984 /**
985 * Insert the method's description here.
986 * Creation date: (02-07-00 19:07:18)
987 * @return clientapp.ClientInfo
988 */
989 public ClientInfo getClientInfo()
990 {
991     if ( clientInfo == null )

```

```

990     {
991         clientInfo = new ClientInfo();
992     }
993     return clientInfo;
994 }
995 /**
996 * Insert the method's description here.
997 * Creation date: (02-07-00 22:53:11)
998 * @return clientapp.GroupManagementClient
999 */
1000 public GroupManagementClient getGroupManagementClient() {
1001     if (groupManagementClient == null)
1002     {
1003         groupManagementClient = new GroupManagementClient();
1004         groupManagementClient.setName( ClientInfo.getPreferences().getProperty( ClientConstants.getNAME()
1005             ));
1006     }
1007     return groupManagementClient;
1008 }
1009 /**
1010 * Insert the method's description here.
1011 * Creation date: (09-10-00 12:53:32)
1012 * @param ce com.sun.media.jsdt.event.ConnectionEvent
1013 */
1014 public void groupManagementConnectionFailed(ConnectionEvent ce)
1015 {
1016     owner.initMessageArea( 1, 1, 1, ClientConstants.getConnection_FAILURE( ce.toString() ) );
1017 }
1018 /**
1019 * Handle the different subclasses of JSDTEexception, i.e. the different types of exception that JSDE
1020 * actions may throw,
1021 * by issuing appropriate messages and taking corrective or conclusive action.
1022 * Creation date: (08-08-00 16:34:28)
1023 * @param exception com.sun.media.jsdt.JSDTEexception
1024 */
1025 public void handleJSDTEexception(JSDTEexception exception)
1026 {
1027     String message = (String) ClientConstants.getJSDE_EXCEPTION_MESSAGES_TABLE().get( exception.getClass()
1028         ().toString() );
1029     message += " (" + exception.toString() + ")";
1030     System.err.println( message );
1031     owner.initMessageArea( 1, 1, 1, message );
1032 }
1033 /**
1034 * Inflate the contents of a compressed document.
1035 * Creation date: (08-07-00 01:56:27)
1036 */
1037 public byte[] inflateDocument( CompressedDocumentPackage compressedDocumentPackage )
1038 {
1039     byte uncompressedData[] = new byte[ compressedDocumentPackage.getUncompressedLength() ];
1040     try
1041     {
1042        Inflater inflater = new Inflater();
1043         inflater.setInput( compressedDocumentPackage.getCompressedContents() );
1044         int inflatedSize = inflater.inflate( uncompressedData );
1045     }
1046     catch ( DataFormatException DFE )
1047     {
1048         System.err.println( DFE.toString() );
1049         System.exit( 0 );
1050     }
1051     // System.out.println( "Deflated: " + compressedDocumentPackage.getCompressedLength() + " Inflated
1052     : " + compressedDocumentPackage.getUncompressedLength() );
1053     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 ( JSDTEException JSDTE )
     {
1076         handleJSDTEException( JSDTE );
     }
1078 }
1079 /**
1080     * Inform other group members that the panorama has changed and must be updated.
     * Creation date: (31-08-00 19:11:21)
1082 */
1083 public void informGroupMembersOfPanoramaUpdate()
1084 {
1085     if ( getClientInfo().isLocalMode() == false )
1086     {
1087         urlString url;
1088         Session session;
1089         Channel groupChannel = null;
1090         try
     {
1091             url = getClientInfo().getActiveGroup().getSessionURL();
             session = SessionFactory.createSession( getGroupManagementClient(), url, false );
1093             groupChannel = session.createChannel( getGroupManagementClient(), getClientInfo().
                 getActiveGroup().getWorkGroupChannelID(), true, true, false );
1094
1095             DataPackage outgoingPackage = new DataPackage();
             outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
                 getUPDATE_PANORAMA() );
1096             groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ) );
1097
1098         }
     }
1099     catch ( JSDTEException JSDTE )
1100     {
1101         handleJSDTEException( JSDTE );
     }
1102     }
1103 }
1104 }
1105 }
1106 }
1107 /**
1108     * Notify the server that this client has terminated.
     * Creation date: (01-11-00 01:16:44)
1110 */
1111 public void informServerOfExit()

```

```

1112 {
1113     DataPackage outgoingPackage = new DataPackage();
1114
1115     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1116         getClientLEFT() );
1117
1118     try
1119     {
1120         if ( getClientInfo().getGroupManagementChannel() != null )
1121         {
1122             getClientInfo().getGroupManagementChannel().sendToClient(
1123                 getGroupManagementClient(),
1124                 ClientInfo.getPreferences().getProperty( ClientConstants.getServer_NAME() ),
1125                 new Data( outgoingPackage ) );
1126         }
1127     }
1128     catch ( JSDTEException JSDTE )
1129     {
1130         handleJSDTEException( JSDTE );
1131     }
1132 }
1133 /**
1134 * Inform the server that the client has exited the group to which it belonged, if any.
1135 * Creation date: (31-10-00 23:55:01)
1136 */
1137 public void informServerOfGroupExit()
1138 {
1139     if ( getClientInfo().getActiveGroup() != null )
1140     {
1141         DataPackage outgoingPackage = new DataPackage();
1142
1143         outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1144             getClientLEFT_GROUP() );
1145         outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), getClientInfo().
1146             getActiveGroup().getWorkGroupID() );
1147
1148         try
1149         {
1150             if ( getClientInfo().getGroupManagementChannel() != null )
1151             {
1152                 getClientInfo().getGroupManagementChannel().sendToClient(
1153                     getGroupManagementClient(),
1154                     ClientInfo.getPreferences().getProperty( ClientConstants.getServer_NAME() ),
1155                     new Data( outgoingPackage ) );
1156             }
1157         }
1158     }
1159 /**
1160 * Inform the server that the client has added one or more documents to the panorama.
1161 * Creation date: (04-10-00 11:39:50)
1162 * @param docPath java.lang.String
1163 */
1164 public void informServerOfNewDocument(String docPath)
1165 {
1166     DataPackage outgoingPackage = new DataPackage();
1167
1168     outgoingPackage.setContents( docPath );
1169     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1170         getClientADDED_DOCUMENTS() );
1171     outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ClientInfo.getPreferences
1172         ().getProperty( ClientConstants.getAUTHOR_NAME() ) );
1173
1174     try
1175     {

```

```

1174     if ( getClientInfo().getGroupManagementChannel() != null )
1175     {
1176         getClientInfo().getGroupManagementChannel().sendToClient(
1177             getGroupManagementClient(),
1178             ClientInfo.getPreferences().getProperty( ClientConstants.getServer_name() ),
1179             new Data( outgoingPackage ) );
1180     }
1181     catch ( JSDTEException JSDTE )
1182     {
1183         handleJSDTEException( JSDTE );
1184     }
1185 }
1186 /**
1187 * 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)
1189 * @param workGroup peerviewmisc.WorkGroup
1190 */
1191 public void informUserOfGroupJoin(WorkGroup workGroup)
1192 {
1193     // if this client is the only group member
1194     if ( workGroup.getParticipants().size() <= 1 )
1195     {
1196         owner.initMessageArea( 1, 1, 1, ClientConstants.getClient_joined_group_message( workGroup ) );
1197     }
1198     else
1199     {
1200         owner.initMessageArea( 1, 1, 1, ClientConstants.getClient_joined_group_with_members_message(
1201             workGroup ) );
1202     }
1203 }
1204 /**
1205 * Display a notification message that the client is operating in local mode, i.e disconnected from the
1206 * server.
1207 * Creation date: (10-08-00 04:34:38)
1208 */
1209 public void informUserOfLocalMode()
1210 {
1211     if ( getClientInfo().isLocalMode() == false )
1212     {
1213         MessageBox messageBox = new MessageBox();
1214
1215         messageBox.setText( ClientConstants.getLocal_mode_message() );
1216         messageBox.show();
1217         getClientInfo().setLocalMode( true );
1218     }
1219 }
1220 /**
1221 * Insert the method's description here.
1222 * Creation date: (04-07-00 17:02:27)
1223 */
1224 public void initialize()
1225 {
1226     clearTopicTree();
1227 }
1228 /**
1229 * Leave the group management communication channel.
1230 * Creation date: (20-10-00 15:43:39)
1231 */
1232 public void leaveGroupManagement()
1233 {
1234     try
1235     {
1236         if ( getClientInfo().getGroupManagementChannel() != null )
1237         {
1238             getClientInfo().getGroupManagementChannel().leave( getGroupManagementClient() );

```

```

1238     }
1239     if ( getClientInfo().getGroupManagementSession() != null )
1240     {
1241         getClientInfo().getGroupManagementSession().leave( getGroupManagementClient() );
1242     }
1243     catch ( JSDTEException JSDTE )
1244     {
1245         handleJSDTEException( JSDTE );
1246     }
1247 }
1248 /**
1249 * Perform termination housekeeping such as updating preferences and
1250 * sundry other chores.
1251 * Creation date: (09-08-00 23:23:16)
1252 */
1253 public void prepareForTermination()
1254 {
1255     java.awt.Dimension size = owner.getSize();
1256     java.awt.Point location = owner.getLocation();

1257     ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_HORIZONTAL_SIZE(),
1258         String.valueOf( (int)size.getWidth() ) );
1259     ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_VERTICAL_SIZE(),
1260         String.valueOf( (int)size.getHeight() ) );
1261     ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_X_COORDINATE(),
1262         String.valueOf( (int)location.getX() ) );
1263     ClientInfo.getPreferences().setProperty( ClientConstants.getFRAME_Y_COORDINATE(),
1264         String.valueOf( (int)location.getY() ) );
1265     if ( getClientInfo().isLocalMode() == false )
1266     {
1267         ClientInfo.getPreferences().setProperty( ClientConstants.getActiveGroup_UPON_EXIT(),
1268             getClientInfo().getActiveGroup().getWorkGroupID() );
1269     }
1270     getClientInfo().writePreferences();
1271     writeDocumentList();
1272     leaveGroupManagement();
1273     owner.getPanoramaPanel().getUpdateTimer().stop();
1274     try
1275     {
1276         System.runFinalization();
1277         Runtime.getRuntime().exit(0);
1278     }
1279     catch ( Exception E )
1280     {
1281         System.out.println( ClientConstants.getCOULD_NOT_EXIT_PROPERLY() );
1282     }
1283 }
1284 /**
1285 * Creation date: (14-06-00 22:34:24)
1286 * @param newfilenames java.lang.File[]
1287 */
1288 public byte[] readDocument(File newFile) throws IOException
1289 {
1290     DataInputStream input;
1291     byte rawFileContents[];
1292     // fetch contents of file as raw array of bytes
1293     rawFileContents = new byte[ (int)newFile.length() ];
1294     try
1295     {
1296         input = new DataInputStream( new FileInputStream(newFile) );
1297         input.read( rawFileContents );
1298     }
1299     catch ( IOException e )
1300     {
1301         System.err.println( ClientConstants.failedToAccessDocument( newFile.toString() ) );
1302         System.exit(1); //SKALE NDRES: fejlmeldelse øbr fremkomme i status area i client window
1303     }

```

```

        return rawFileContents;
1304    }
1305    /**
1306     * Read the list of documents from disk and insert the file names it contains into a vector.
1307     * Creation date: (06-09-00 21:17:15)
1308     */
1309    public java.util.Vector readDocumentList()
1310    {
1311        java.util.Vector docs = null;
1312        try
1313        {
1314            FileInputStream fis = new FileInputStream( ClientConstants.getDOCUMENT_LIST_FILENAME() );
1315            ObjectInputStream ois = new ObjectInputStream( fis );
1316            try
1317            {
1318                docs = (java.util.Vector) ois.readObject();
1319            }
1320            catch ( ClassNotFoundException CNFE )
1321            {
1322                System.err.println( CNFE.toString() );
1323            }
1324            ois.close();
1325        }
1326        catch ( IOException ioe )
1327        {
1328            System.err.println( ClientConstants.getCOULD_NOT_READ_DOCUMENT_LIST() );
1329            owner.initMessageArea( 1, 1, 1, ClientConstants.getCOULD_NOT_READ_DOCUMENT_LIST() );
1330        }
1331        return docs;
1332    }
1333    /**
1334     * Remove from the panorama all documents originating from other members of the group to which this
1335     * client currently belongs.
1336
1337     * Creation date: (15-07-00 16:23:13)
1338     * @param senderName java.lang.String
1339     * @param fullyQualifiedDocumentName java.lang.String
1340     */
1341    public void removeDistributedDocument(String authorName, String senderName, String documentPath)
1342    {
1343        owner.initMessageArea( 1, 1, 1, ClientConstants.getREMOVING_DISTRIBUTED_DOCUMENT( documentPath,
1344            authorName ) );
1345        owner.getPanoramaPanel().removeGlobalDocument( senderName, documentPath );
1346        owner.getPanoramaPanel().revalidate();
1347        owner.getPanoramaPanel().repaint();
1348    }
1349    /**
1350     * Remove all distributed documents from the local panorama and clear the hashtable indexing them.
1351     * Creation date: (26-07-00 02:54:40)
1352     */
1353    public void removeDistributedDocumentsFromPanorama()
1354    {
1355        ( new RemoveDistributedDocumentsFromPanorama() ).start();
1356        owner.getPanoramaPanel().zoomToOverview();
1357    }
1358    /**
1359     * Launch a removal process for a set of documents in a separate thread (to allow concurrent updating of
1360     * the GUI).
1361     * Creation date: (03-10-00 11:23:07)
1362     * @param removables java.io.File[]
1363     */
1364    public void removeDocuments(File[] removables)
1365    {
1366        ( new RemoveDocuments( removables ) ).start();
1367    }
1368}

```

```

1366 * Submit a request to the server that it removes a specific group from the group directory.
1367 * Creation date: (26-07-00 18:57:18)
1368 * @param workGroup peerviewmisc.WorkGroup
1369 */
1370 public void removeWorkGroup(WorkGroup workGroup)
1371 {
1372     DataPackage outgoingPackage = new DataPackage();
1373
1374     outgoingPackage.setContents( workGroup );
1375     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1376         getRequest_REMOVE_GROUP() );
1377     try
1378     {
1379         if ( getClientInfo().getGroupManagementChannel() != null )
1380         {
1381             getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1382                 ClientInfo.getPreferences().getProperty( ClientConstants.
1383                     getSERVER_NAME() ),
1384                 new Data( outgoingPackage ) );
1385         }
1386     }
1387     catch ( JSDTEException JSDTE )
1388     {
1389         handleJSDTEException( JSDTE );
1390     }
1391
1392 /**
1393 * Request copy of documents from all other members of active group.
1394 * Creation date: (08-07-00 20:13:19)
1395 */
1396 public void requestDocuments()
1397 {
1398     if ( getClientInfo().isLocalMode() == false )
1399     {
1400         WorkGroup activeGroup = getClientInfo().getActiveGroup();
1401         if ( activeGroup.getDocuments().size() > 0 )
1402         {
1403             // System.out.println( "Requested documents" );
1404             //owner.initMessageArea( 1, activeGroup.getDocuments().size() - getClientInfo().getLocalFiles()
1405             .size(), 1,
1406             // ClientConstants.getRequestING_DOCUMENTS_MESSAGE( activeGroup.getDocuments().
1407             size() - getClientInfo().getLocalFiles().size() );
1408
1408         DataPackage outgoingPackage = new DataPackage();
1409         outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1410             getRequest_DOCUMENTS() );
1411         outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ClientInfo.
1412             getPreferences().getProperty( ClientConstants.getAUTHOR_NAME() ) );
1413         try
1414         {
1415             urlString url = activeGroup.getSessionURL();
1416             Session session = SessionFactory.createSession( getGroupManagementClient(), url, false );
1417             Channel groupChannel = session.createChannel( getGroupManagementClient(), activeGroup.
1418                 getWorkGroupChannelID(), true, true, false );
1419
1420             groupChannel.sendToOthers( getGroupManagementClient(), new Data( outgoingPackage ) );
1421         }
1422         catch ( JSDTEException JSDTE )
1423         {
1424             handleJSDTEException( JSDTE );
1425         }
1426     }
1427 }
1428 /**

```

```

1426 * Request the directory of groups from the server. Assumes that group management is up and running.
1427 * The directory will be received by the groupManagementClient data member.
1428 * Creation date: (04-07-00 16:27:33)
1429 */
1430 public void requestGroupsFromServer()
1431 {
1432     DataPackage outgoingPackage = new DataPackage();
1433     outgoingPackage.setContents( null );
1434     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1435         getREQUEST_GROUP_DIRECTORY() );
1436     try
1437     {
1438         if ( getClientInfo().getGroupManagementChannel() != null )
1439         {
1440             for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i
1441                 ++ )
1442                 {
1443                     // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
1444                     getClientInfo().getGroupManagementChannel().sendToClient( groupManagementClient,
1445                         ClientInfo.getPreferences().getProperty( ClientConstants.
1446                             SERVER_NAME() ),
1447                         new Data( outgoingPackage ) );
1448                 }
1449             }
1450         catch ( JSDTEException JSDTE )
1451         {
1452             handleJSDTEException( JSDTE );
1453         }
1454     }
1455 /**
1456 * Request a specific message from the message database maintained by the server.
1457 * Creation date: (10-07-00 02:11:23)
1458 * @param messageID java.lang.String
1459 */
1460 public void requestMessageFromServer(String messageID)
1461 {
1462     DataPackage outgoingPackage = new DataPackage();
1463     outgoingPackage.setContents( null );
1464     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1465         getREQUEST_MESSAGE() );
1466     outgoingPackage.getProperties().setProperty( DataPackage.getMessage_ID(), messageID );
1467     try
1468     {
1469         if ( getClientInfo().getGroupManagementChannel() != null )
1470         {
1471             for ( int i = 0; i < getClientInfo().getGroupManagementChannel().listConsumerNames().length; i
1472                 ++ )
1473                 {
1474                     // System.out.println( getClientInfo().getGroupManagementChannel().listConsumerNames()[i] );
1475                     clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1476                         ClientInfo.getPreferences().getProperty( ClientConstants.
1477                             SERVER_NAME() ),
1478                         new Data( outgoingPackage ) );
1479                 }
1480             }
1481         catch ( JSDTEException JSDTE )
1482         {
1483             handleJSDTEException( JSDTE );
1484         }
1485     }
1486 /**
1487 * Request that the server submit the current topic tree for the document designated by documentName.
1488 * Creation date: (09-07-00 18:35:18)

```

```

1486 * @param documentName java.lang.String
1487 */
1488 public void requestTopicTree(String documentName)
1489 {
1490     System.out.println( "Inde_ ui requestTopicTree" );
1491     DataPackage outgoingPackage = new DataPackage();
1492     outgoingPackage.setContents( documentName );
1493     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1494         getRequest_TOPIC_TREE() );
1495     try
1496     {
1497         if ( getClientInfo().getGroupManagementChannel() != null )
1498         {
1499             getClientInfo().getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1500                 ClientInfo.getPreferences().getProperty( ClientConstants.
1501                     SERVER_NAME() ),
1502                 new Data( outgoingPackage ) );
1503         }
1504     }
1505     catch ( JSDTEException JSDTE )
1506     {
1507         handleJSDTEException( JSDTE );
1508     }
1509 }
1510 /**
1511 * Notify the server that the client added a message.
1512 * Creation date: (11-07-00 21:23:22)
1513 * @param newMessage peerviewmisc.Message
1514 */
1515 public void sendMessage(Message newMessage)
1516 {
1517     DataPackage outgoingPackage = new DataPackage();
1518     outgoingPackage.setContents( newMessage );
1519     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1520         getClient_ADDED_MESSAGE() );
1521     try
1522     {
1523         if ( getClientInfo().getGroupManagementChannel() != null )
1524         {
1525             clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1526                 ClientInfo.getPreferences().getProperty( ClientConstants.
1527                     SERVER_NAME() ),
1528                 new Data( outgoingPackage ) );
1529         }
1530     }
1531     catch ( JSDTEException JSDTE )
1532     {
1533         handleJSDTEException( JSDTE );
1534     }
1535 }
1536 /**
1537 * Submit a request to the server that a specific message be deleted.
1538 * Creation date: (12-07-00 12:05:06)
1539 * @param messageID java.lang.String
1540 */
1541 public void sendMessageDeletion(String messageID)
1542 {
1543     DataPackage outgoingPackage = new DataPackage();
1544     outgoingPackage.setContents( messageID );
1545     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1546         getDELETE_MESSAGE() );
1547     try
1548     {
1549         if ( getClientInfo().getGroupManagementChannel() != null )

```

```

1546     {
1547         clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1548                                         ClientInfo.getPreferences().getProperty( ClientConstants.
1549                                         SERVER_NAME() ),
1550                                         new Data( outgoingPackage ) );
1551     }
1552     catch ( JSDTException JSDTE )
1553     {
1554         handleJSDTException( JSDTE );
1555     }
1556 }
1557 /**
1558 * Submit an updated topic tree to the server for inclusion in its topic tree database.
1559 * Creation date: (11-07-00 21:23:37)
1560 */
1561 public void sendTopicTreeUpdate( DefaultTreeModel newTree )
1562 {
1563     DataPackage outgoingPackage = new DataPackage();
1564     outgoingPackage.setContents( newTree );
1565     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ClientConstants.
1566                                         getTOPIC_TREE_UPDATE() );
1567     outgoingPackage.getProperties().setProperty( DataPackage.getDOC PATH(), getClientInfo().
1568                                         getCurrentDocumentName() );
1569     try
1570     {
1571         if ( getClientInfo().getGroupManagementChannel() != null )
1572         {
1573             clientInfo.getGroupManagementChannel().sendToClient( getGroupManagementClient(),
1574                                         ClientInfo.getPreferences().getProperty( ClientConstants.
1575                                         SERVER_NAME() ),
1576                                         new Data( outgoingPackage ) );
1577         }
1578     }
1579     catch ( JSDTException JSDTE )
1580     {
1581         handleJSDTException( JSDTE );
1582     }
1583 /**
1584 * Insert the method's description here.
1585 * Creation date: (02-07-00 19:07:18)
1586 * @param newClientInfo clientapp.ClientInfo
1587 */
1588 public void setClientInfo(ClientInfo newClientInfo) {
1589     clientInfo = newClientInfo;
1590 }
1591 /**
1592 * Update the text area for message display in the discussion panel at the bottom of the client main
1593 * window.
1594 * Creation date: (10-07-00 02:30:47)
1595 * @param newMessage peerviewmisc.Message
1596 */
1597 public void setDisplayedMessage(Message newMessage)
1598 {
1599     owner.setMessage( newMessage );
1600     getClientInfo().setCurrentMessage( newMessage );
1601 }
1602 /**
1603 * Insert the method's description here.
1604 * Creation date: (02-07-00 22:53:11)
1605 * @param newGroupManagementClient clientapp.GroupManagementClient
1606 */
1607 public void setGroupManagementClient(GroupManagementClient newGroupManagementClient) {
1608     groupManagementClient = newGroupManagementClient;

```

```

    }

1608 /**
 * Insert the method's description here.
1609 * Creation date: (09-07-00 22:48:06)
1610 * @param newTree javax.swing.tree.DefaultTreeModel
1611 * @param documentName java.lang.String
1612 */
1613 public void setTopicTree(DefaultTreeModel newTree, String documentName)
1614 {
1615     setDisplayedMessage( null );
1616     getClientInfo().setCurrentDocumentName( documentName );
1617     owner.setTopicTree( newTree );
1618 }
1619 /**
 * Update one of the distributed documents shown in the panorama.
1620 * Creation date: (21-08-00 17:30:56)
1621 * @param documentName java.lang.String
1622 * @param senderName java.lang.String
1623 * @param compressedDifferenceBitmapPackage peerviewmisc.CompressedDocumentPackage
1624 */
1625 public void updateDistributedDocument(String documentName, String senderName, CompressedDocumentPackage
1626                                         compressedDifferenceBitmapPackage)
1627 {
1628     owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATING_DISTRIBUTED_DOCUMENT( documentName ) );
1629     byte rawFileContents[] = inflateDocument( compressedDifferenceBitmapPackage );
1630     owner.getPanoramaPanel().updateDistributedDocument( senderName, documentName, rawFileContents );
1631 }
1632 /**
 * Extract difference bitmap by XORing old and new file contents. Then submit the result as a deflated,
1633 * i.e. gzipped,
1634 * data package to other members, if any, of active group.
1635 * Creation date: (21-08-00 16:01:13)
1636 * @param path java.lang.String
1637 * @param clientName java.lang.String
1638 * @param oldRawFileContents byte[]
1639 * @param newRawFileContents byte[]
1640 */
1641 public void updateDocument(String path, String clientName, byte[] oldRawFileContents, byte[]
1642                             newRawFileContents)
1643 {
1644     byte differenceBitmap[] = computeBitmapCombination( oldRawFileContents, newRawFileContents );
1645     CompressedDocumentPackage compressedDifferenceBitmapPackage = deflateDocument( differenceBitmap );
1646     broadcastDocumentUpdate( path, compressedDifferenceBitmapPackage );
1647 }
1648 /**
 * Insert the method's description here.
1649 * Creation date: (04-08-00 15:50:41)
1650 * @param newDirectory java.util.Hashtable
1651 */
1652 public void updateGroupDirectory(Hashtable newDirectory)
1653 {
1654     getClientInfo().setGroups( newDirectory );
1655     owner.getGroupDirectory1().updateTable();
1656     if ( getClientInfo().getActiveGroup() != null )
1657     {
1658         getClientInfo(). setActiveGroup( (WorkGroup) newDirectory.get( getClientInfo().getActiveGroup().
1659                                         getWorkGroupID() ) );
1660     }
1661 }
1662 /**
 * Insert the method's description here.
1663 * Creation date: (14-06-00 23:07:20)
1664 */
1665 public void updatePanorama()
1666 {
1667     owner.updateVisibleDocumentsBox();

```

```

1670     // System.out.println("updatePanorama");
1671     owner.getPanoramaPanel().validateLayout();
1672 }
1673 /**
1674 * Insert the method's description here.
1675 * Creation date: (31-08-00 02:14:49)
1676 */
1677 public void updatePanoramaByRequest()
1678 {
1679     owner.initMessageArea( 1, 1, 1, ClientConstants.getUPDATE_PANORAMA_BY_REQUEST_MESSAGE() );
1680     updatePanorama();
1681 }
1682 /**
1683 * Insert the method's description here.
1684 * Creation date: (27-08-00 21:21:06)
1685 * @param fullyQualifiedDocumentName java.lang.String
1686 * @param treeUpdate javax.swing.tree.DefaultTreeModel
1687 */
1688 public void updateTopicTree(String senderName, String documentName, DefaultTreeModel treeUpdate)
1689 {
1690     String fqName = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
1691     if ( getClientInfo().getCurrentDocumentName().equals( fqName ) )
1692     {
1693         setTopicTree( treeUpdate, fqName );
1694     }
1695 }
1696 /**
1697 * Test connection to group management and notify user if it isn't active.
1698 * Creation date: (09-10-00 18:57:46)
1699 * @return boolean
1700 */
1701 public boolean verifyGroupConnectionManagement()
1702 {
1703     if ( getClientInfo().getGroupManagementSession() == null )
1704     {
1705         owner.initMessageArea( 1, 1, 1, ClientConstants.getNOT_CONNECTED_TO_SERVER() );
1706         return false;
1707     }
1708     else
1709     {
1710         return true;
1711     }
1712 }
1713 /**
1714 * Extract a list of documents displayed in the panorama and write it to disk. This can then be restored
1715 * automatically
1716 * the next time the client is started.
1717 * Creation date: (06-09-00 21:48:43)
1718 */
1719 public void writeDocumentList()
1720 {
1721     try
1722     {
1723         FileOutputStream fos = new FileOutputStream( ClientConstants.getDOCUMENT_LIST_FILENAME() );
1724         ObjectOutputStream oos = new ObjectOutputStream( fos );
1725
1726         java.util.Vector docs = new java.util.Vector();
1727         Iterator iterator = owner.getPanoramaPanel().getLocalDocuments().values().iterator();
1728         while ( iterator.hasNext() )
1729         {
1730             edu.umd.cs.jazz.ZVisualLeaf leaf = (edu.umd.cs.jazz.ZVisualLeaf) iterator.next();
1731             DocPackage docPackage = new DocPackage( (String) leaf.getClientProperty( ClientConstants.
1732                 getDOCUMENT_NAME() ),
1733                                         new java.awt.Dimension( (int) leaf.getVisualComponent().getBounds() .
1734                                         getMaxX(),

```

```

1732                               (int) leaf.getVisualComponent().getBounds().getMaxY
1733                               ( ) ,
1734                               new java.awt.Dimension( (int) leaf.editor().getTransformGroup().
1735                               getTranslateX(),
1736                               (int) leaf.editor().getTransformGroup().
1737                               getTranslateY() )
1738 );
1739 docPackage.setNodeLocked( (String) leaf.getClientProperty( ClientConstants.getNodeLocked() ) );
1740 docs.add( docPackage );
1741 }
1742 oos.writeObject( docs );
1743 oos.flush();
1744 oos.close();
1745 }
1746 catch ( IOException ioe )
1747 {
1748     System.err.println( ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
1749     owner.initMessageArea( 1, 1, 1, ClientConstants.getCOULD_NOT_WRITE_DOCUMENT_LIST() );
1750 }
1751 }
1752 }

```

Listing C.6: ClientConstants.java

```

package clientapp;
2

4 import java.awt.Color;
5 import java.util.Hashtable;
6 import edu.umd.cs.jazz.*;
7 import edu.umd.cs.jazz.component.*;
8 import edu.umd.cs.jazz.event.*;
9 import edu.umd.cs.jazz.io.*;
10 import edu.umd.cs.jazz.util.*;
11 import java.util.Properties;
12 import peerviewmisc.SharedConstants;
13 import java.util.Random;
14
15 /**
16 * This class contains constants specific to the client application as well as those derived from the {
17 * @link #peerviewmisc.SharedConstants SharedConstants} class.
18 * Creation date: (15-06-00 22:57:59)
19 * @author:
20 */
21 public class ClientConstants extends SharedConstants {

22     private final static Color BACKGROUND_COLOUR = new Color(200, 200, 200);
23     public final static java.lang.String PROCESSING_DOCUMENTS = "Adding\u00a9documents\u00a9to\u00a9the\u00a9panorama";
24     public final static java.lang.String MESSAGE_AREA_DEFAULT_STRING = "Message\u00a9area:\u00a9double-click\u00a9here\u00a9
25             to\u00a9open\u00a9in\u00a9separate\u00a9window";
26     private final static int UPDATE_INTERVAL_IN_MILLISECS = 1;
27     private final static java.lang.String RETRIEVING_DOCUMENTS = "Adding\u00a9documents\u00a9to\u00a9panorama";

28     private final static java.lang.String PLAIN_TEXT = "text/plain";
29     private final static java.lang.String HTML = "text/html";
30     private final static java.lang.String RTF = "text/rtf";
31     private final static java.lang.String BITMAP = "bitmap";
32     private final static java.lang.String ALL_FILES = "All\u00a9files";
33     private final static java.lang.String[][] FileTypes = { {"*.txt", "Plain\u00a9text\u00a9(.txt)", PLAIN_TEXT},
34                                         {"*.html", "HTML\u00a9(.html,\u00a9.htm)", HTML},
35                                         {"*.rtf", "RTF\u00a9(.rtf)", RTF},
36                                         {"*.cpp", "C++\u00a9source\u00a9(.cpp,\u00a9.cc)", PLAIN_TEXT},
37                                         {"*.h", "C++\u00a9header\u00a9(.h)", PLAIN_TEXT},
38                                         {"*.java", "Java\u00a9source\u00a9(.java)", PLAIN_TEXT},
39                                         {"*.gif", "GIF\u00a9(.gif)", BITMAP},
40                                         {"*.jpg", "JPG\u00a9(.jpg)", BITMAP} };

```

```

42     private final static java.lang.String[][] FILE_TYPE_SYNONYM_TABLE = { {".txt"},  

43                               {"html", ".htm"},  

44                               {"rtf"},  

45                               {"cpp", ".cc"},  

46                               {"h"},  

47                               {"java"},  

48                               {"gif"},  

49                               {"jpg"}  

50   };  

51  

52   private final static java.lang.String OUT_OF_BOUNDS_EXCEPTION_TEXT = "Could not add all documents to  

53   panorama due to layout restrictions";  

54   private final static int FILE_EXTENSION_INDEX = 0;  

55   private final static int FILE_TYPE_DESCRIPTION_INDEX = 1;  

56   private final static int TYPE_INDEX = 2;  

57  

58   private final static java.lang.String REMOVING_DOCUMENTS = "Removing documents from panorama";  

59  

60   private final static java.lang.String[] DISPLAY_QUALITIES = { {"Low", String.valueOf(  

61                           ZDrawingSurface.RENDER_QUALITY_LOW)},  

62                           {"Medium", String.valueOf(ZDrawingSurface.  

63                                         RENDER_QUALITY_MEDIUM)},  

64                           {"High", String.valueOf(ZDrawingSurface.  

65                                         RENDER_QUALITY_HIGH)} };  

65  

66   public final static java.lang.String GRID_LAYOUT = "Grid_layout";  

67  

68   public final static java.lang.String UPDATE_INTERVAL = "Update_interval";  

69   private final static java.lang.String NAME = "Unique_client_ID";  

70   private final static java.lang.String SIGNATURE = "Signature";  

71   private final static java.lang.String DISPLAY_QUALITY = "Display_quality";  

72   private final static java.lang.String LAYOUT_SCHEME = "Layout_scheme";  

73   public final static java.lang.String LOCAL_HOST = "localhost";  

74   public final static java.lang.String ANIMATION_DURATION_IN_MILLISECS = "Animation_duration";  

75   private final static java.lang.String AUTHOR_NAME = "Author_name";  

76   private final static java.lang.String DEFAULT_DOCUMENT_WIDTH = "Default_document_width";  

77   private final static java.lang.String DEFAULT_DOCUMENT_HEIGHT = "Default_document_height";  

78   private final static java.lang.String DEFAULT_HORIZONTAL_DOCUMENT_SPACING = "Default_horizontal_  

79   document_spacing";  

80   private final static java.lang.String DEFAULT_VERTICAL_DOCUMENT_SPACING = "Default_vertical_document_  

81   spacing";  

82   private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS = "Grid_layout_default_  

83   number_of_rows";  

84   private final static java.lang.String GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS = "Grid_layout_default_  

85   number_of_columns";  

86   private final static java.lang.String FRAME_HORIZONTAL_SIZE = "Frame_horizontal_size";  

87   private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame_vertical_size";  

88   private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";  

89   private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";  

90   private final static java.lang.String FRAME_X_COORDINATE = "Frame_x_coordinate";  

91   private final static java.lang.String FRAME_Y_COORDINATE = "Frame_y_coordinate";  

92   private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";  

93   private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";  

94   private final static java.lang.String ACTIVE_GROUP_UPON_EXIT = "Active_group_upon_exit";  

95   private final static java.lang.String DEFAULT_GROUP_ID = "";  

96   private final static java.lang.String ZOOM_TO_OVERVIEW_ON_UPDATE = "Zoom_to_overview_on_update";  

97   private final static java.lang.String ZOOM_TO_OVERVIEW = "zoom_to_overview_on_update";  

98   private final static java.lang.String DO_NOT_ZOOM_TO_OVERVIEW = "do_not_zoom_to_overview";  

99   private final static java.lang.String SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX = "Show_full_  

100  pathnames_in_visible_documents_box";  

101  public final static java.lang.String TRUE = "true";  

102  public final static java.lang.String FALSE = "false";  

103  private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH = "Default_grid_layout_  

104  document_width";

```

```

96     private final static java.lang.String DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT = "Default_grid_layout_
97         document_height";
98     private final static java.lang.String DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING = "Default_grid_layout_
99         horizontal_spacing";
100    private final static java.lang.String DEFAULT_GRID_LAYOUT_VERTICAL_SPACING = "Default_grid_layout_
101        vertical_spacing";
102    private final static java.lang.String LOAD_DOCUMENT_LIST_ON_STARTUP = "Load_document_list_on_startup"
103        ;
104    private final static java.lang.String[][] DEFAULT_PREFERENCES = { {UPDATE_INTERVAL, "5"}, 
105        {NAME, DEFAULT_CLIENT_NAME}, 
106        {AUTHOR_NAME, "?"}, 
107        {SIGNATURE, ""}, 
108        {DISPLAY_QUALITY, "High"}, 
109        {LAYOUT_SCHEME, GRID_LAYOUT}, 
110        {SERVER_NAME, DEFAULT_SERVER_NAME}, 
111        {SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT)}, 
112        {CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE}, 
113        {ANIMATION_DURATION_IN_MILLISECS, "1000"}, 
114        {DEFAULT_DOCUMENT_WIDTH, "400"}, 
115        {DEFAULT_DOCUMENT_HEIGHT, "500"}, 
116        {DEFAULT_HORIZONTAL_DOCUMENT_SPACING, "500"}, 
117        {DEFAULT_VERTICAL_DOCUMENT_SPACING, "500"}, 
118        {GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS, "10"}, 
119        {GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS, "10"}, 
120        {FRAME_HORIZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE}, 
121        {FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE}, 
122        {FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE}, 
123        {FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE}, 
124        {ACTIVE_GROUP_UPON_EXIT, DEFAULT_GROUP_ID}, 
125        {ZOOM_TO_OVERVIEW_ON_UPDATE, ZOOM_TO_OVERVIEW}, 
126        {getCOMPRESSION_LEVEL(), String.valueOf(
127            getDefaultCompressionLevel())}, 
128        {SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX, FALSE}, 
129        {DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH, "500"}, 
130        {DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT, "600"}, 
131        {DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING, "100"}, 
132        {DEFAULT_GRID_LAYOUT_VERTICAL_SPACING, "100"}, 
133        {LOAD_DOCUMENT_LIST_ON_STARTUP, getTRUE()}, 
134        {getREGISTRY_PORT(), getDefaultRegistryPort()}, 
135        {getAUTOLOCK_NODES(), getFALSE()}, 
136        {getMESSAGE_LOG_FILENAME(), "message.log"}, 
137    };
138
139    private final static java.lang.String PREFERENCES_FILE_NAME = "prefs.ini";
140    private final static java.lang.String[] LAYOUT_SCHEMES = {"Grid_layout"};
141    private final static java.lang.String PROPERTIES_HEADER = "PeerView_client_settings";
142    private final static double DEFAULT_PANORAMA_PERCENTUAL_SIZE = 100;
143
144    private final static int LOCAL_DOCUMENT = 0;
145    public final static java.lang.String DISTRIBUTED_DOCUMENT = "Distributed_document";
146    private final static java.lang.String REQUEST_DOCUMENTS = "Request_documents";
147    private final static java.lang.String DOCUMENT_PATH = "Document_path";
148    private final static java.lang.String NO_TOPIC_TREE_SELECTION_ERROR_MESSAGE = "You must select an
149        item from the topic tree." ;
150    private final static double SCALE_FACTOR_SCALING = 100.0;
151    public final static java.lang.String INVALID_DELETION_ATTEMPTED_MESSAGE = "You can only delete
152        messages that are composed by you and which have not yet been responded to.";
153    public final static java.lang.String MESSAGE_PROPERTIES_NAME_COLUMN_HEADING = "Name";
154    public final static java.lang.String MESSAGE_PROPERTIES_VALUE_COLUMN_HEADING = "Value";
155    private final static int GROUP_DIRECTORY_COLUMN_WIDTH = 250;
156    public final static java.lang.String NO_GROUP_SELECTED_TO_JOIN_MESSAGE = "You must select a group
157        from the list before joining.";
158    private final static java.lang.String GROUP_AT_MAXIMUM_MEMBERS_MESSAGE = "You cannot join the
159        selected group because it has reached its maximum number of participants";
160    public final static java.lang.String ALREADY_JOINED_TO_GROUP_MESSAGE = "You have already joined the
161        selected group.";
```

```

152    public final static java.lang.String BROADCASTING_DOCUMENT_REMOVALS = "Instructing the other group members to expunge your documents from their panoramas.";
153    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.";
155    private final static java.util.Hashtable MEDIA_TYPES = new Hashtable();
156    private final static java.awt.Color RESIZE_RECTANGLE_PEN_COLOUR = Color.green;
157    private final static java.awt.Color ZOOM_VECTOR_COLOUR = Color.green;
158    private final static int DOCUMENT_BORDER_WIDTH = 6;
159    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;
161    private final static int ZOOM_VECTOR_WIDTH = 2;
162    public final static float RENDER_CUTOFF = (float)0.1;
163    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
165    private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.9;
166    private final static java.lang.String CLIPBOARD_NAME = "Peerview clipboard";
167    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.";
169    private final static int RESET_MESSAGE_AREA_DELAY = 7000;
170    private final static java.lang.String CONTENTS_AND_SIGNATURE_DIVIDER = "\n\n\n\n";
171
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 either the server or other group members. To attempt a new connection, please join one of the groups listed in the group directory available from the menubar.";
173    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;
175    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.";
177    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.";
179    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.";
181
182    private final static java.lang.String NOTHING_TO_REMOVE_MESSAGE = "Nothing to remove.";
183    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.";
185    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...";
187    private final static java.lang.String LAST_MODIFIED = "Last modified";
188    private final static java.lang.String DOCUMENT_NAME = "Document name";
189    private final static java.lang.String RAW_BITMAP = "Raw bitmap";
190    private final static java.lang.String DOCUMENT_UPDATE = "Document update";
191    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;
193    private final static int ANIMATION_MIN_SPEED = 300;
194    private final static int UPDATE_NOTIFICATION_THRESHOLD = 15;

```

```

196     private final static java.lang.String VISIBLE_DOCUMENTS_BOX_CAPTION = "Documents\u2014in\u2014panorama";
197     private final static int VISIBLE_DOCUMENTS_ICON_TEXT_GAP = 8;
198     private final static java.lang.String VISIBLE_DOCUMENTS_TAB_SIZE = "uuuuuu";
199     private final static java.lang.String SENDER_NAME = "Sender\u2014name";
200
201     private final static java.lang.String FINISHED_ADDING_DOCUMENTS = "Finished\u2014adding\u2014documents";
202     private final static java.lang.String UPDATE_PANORAMA = "Update\u2014panorama";
203     private final static java.awt.Color VISIBLE_DOCUMENTS_SELECTION_COLOUR = new Color(0,0,80);
204     private final static java.lang.String UPDATE_PANORAMA_BY_REQUEST_MESSAGE = "Updating\u2014the\u2014panorama\u2014by\u2014
205         request\u2014from\u2014other\u2014group\u2014member.";
206     private final static java.awt.Dimension VISIBLE_DOCUMENTS_ITEM_MAX_SIZE = new java.awt.Dimension
207         ( 350, 30 );
208
209     private final static java.lang.String ABOUT_BOX_MESSAGE = "Release\u20140.900\u2014(beta).";
210     private final static int MAX_GRID_LAYOUT_DOCUMENT_HEIGHT = 2000;
211     private final static int MIN_GRID_LAYOUT_DOCUMENT_HEIGHT = 300;
212     private final static int MAX_GRID_LAYOUT_DOCUMENT_WIDTH = 1500;
213     private final static int MIN_GRID_LAYOUT_DOCUMENT_WIDTH = 400;
214     private final static int MAX_GRID_LAYOUT_HORIZONTAL_SPACING = 500;
215     private final static int MIN_GRID_LAYOUT_HORIZONTAL_SPACING = 0;
216     private final static int MIN_GRID_LAYOUT_VERTICAL_SPACING = 0;
217     private final static int MAX_GRID_LAYOUT_VERTICAL_SPACING = 500;
218     private final static java.lang.String DOCUMENT_LIST_FILENAME = "doclist.dat";
219     private final static java.lang.String COULD_NOT_READ_DOCUMENT_LIST = "Unable\u2014to\u2014read\u2014and\u2014process\u2014the\u2014
220         list\u2014of\u2014documents\u2014from\u2014disk.";
221     private final static java.lang.String COULD_NOT_WRITE_DOCUMENT_LIST = "Could\u2014not\u2014write\u2014the\u2014list\u2014of\u2014
222         documents\u2014to\u2014disk.";
223
224     private final static int TEXT_DOCUMENT_THRESHOLD = 1000000;
225
226     private final static java.lang.String NOT_CONNECTED_TO_SERVER = "This\u2014action\u2014cannot\u2014be\u2014completed\u2014
227         because\u2014there\u2014is\u2014currently\u2014no\u2014connection\u2014to\u2014the\u2014server.";
228
229     private final static int MAX_DOCNAME_LENGTH = 75;
230
231     private final static java.lang.String NODE_LOCKED = "Node\u2014locked";
232
233     private final static java.lang.String AUTOLOCK_NODES = "Autolock\u2014nodes";
234
235     private final static java.lang.String MESSAGE_LOG_FILENAME = "Message\u2014log\u2014filename";
236
237     private final static java.lang.String MESSAGE_LOG_WRITE_ERROR = "Could\u2014not\u2014write\u2014the\u2014message\u2014log\u2014to\u2014
238         disk.";
239
240     private final static java.lang.String MESSAGE_LOG_MESSAGE = "Wrote\u2014message\u2014log\u2014to\u2014disk";
241
242     private final static java.lang.String NOTIFYING_SERVER_OF_NAME_IN_USE = "Your\u2014access\u2014to\u2014the\u2014server\u2014is\u2014
243         blocked.\u2014Requesting\u2014a\u2014clean\u2014up...";
244
245     private final static java.lang.String CLIENT_DOCUMENTATION_HELPSET_NAME = "peerview.hs";
246
247     private final static java.lang.String COULD_NOT_INITIALIZE_HELP = "Could\u2014not\u2014initialize\u2014the\u2014help\u2014
248         system.\u2014Online\u2014help\u2014will\u2014not\u2014be\u2014available.";
249
250     private final static java.lang.String CLIENT_HELP_PATH = "help";
251
252     private static java.util.Hashtable FILE_TYPE_SYNONYMS = null;
253
254     /**
255      * ClientMessages constructor comment.
256      */
257     public ClientConstants() {
258         super();
259         initialize();
260     }
261
262     /**
263      * Insert the method's description here.
264      * Creation date: (15-06-00 23:12:13)
265      * @return java.lang.String
266      * @param fileName java.lang.String
267      */
268     static public String failedToAccessDocument(String fileName) {
269         return "An\u2014error\u2014occurred\u2014while\u2014trying\u2014to\u2014access\u2014the\u2014document\u2014with\u2014filename\u2014\""+fileName+"\".+fileName+"\".";
270     }
271
272     /**
273      * Insert the method's description here.
274      * Creation date: (02-09-00 12:29:51)
275      * @return java.lang.String
276      */
277     public final static java.lang.String getABOUT_BOX_MESSAGE() {

```

```

        return ABOUT_BOX_MESSAGE;
254    }
255    /**
256     * Insert the method's description here.
257     * Creation date: (09-08-00 23:05:17)
258     * @return java.lang.String
259     */
260    public final static java.lang.String getACTIVE_GROUP_UPON_EXIT() {
261        return ACTIVE_GROUP_UPON_EXIT;
262    }
263    /**
264     * Insert the method's description here.
265     * Creation date: (19-08-00 20:59:44)
266     * @param documentName java.lang.String
267     * @param senderName java.lang.String
268     */
269    public final static String getADDED_DISTRIBUTED_DOCUMENT_MESSAGE(String documentName, String senderName)
270    {
271        return "Received\u2014" + documentName + "\u2014from\u2014" + senderName;
272    }
273    /**
274     * Insert the method's description here.
275     * Creation date: (05-09-00 14:50:29)
276     * @return java.lang.String
277     * @param authorName java.lang.String
278     * @param numDoks int
279     */
280    public final static String getADDING_DISTRIBUTED_DOCUMENTS(String authorName, int numDoks)
281    {
282        return "Adding\u2014" + numDoks + "\u2014documents\u2014received\u2014from\u2014" + authorName;
283    }
284    /**
285     * Insert the method's description here.
286     * Creation date: (05-10-00 23:05:28)
287     * @return java.lang.String
288     */
289    public final static java.lang.String getAll_FILES() {
290        return ALL_FILES;
291    }
292    /**
293     * Insert the method's description here.
294     * Creation date: (25-07-00 15:52:24)
295     * @return java.lang.String
296     */
297    public final static java.lang.String getALREADY_JOINED_TO_GROUP_MESSAGE() {
298        return ALREADY_JOINED_TO_GROUP_MESSAGE;
299    }
300    /**
301     * Insert the method's description here.
302     * Creation date: (13-07-00 13:08:57)
303     * @return java.lang.String
304     */
305    public final static java.lang.String getANIMATION_DURATION_IN_MILLISECS() {
306        return ANIMATION_DURATION_IN_MILLISECS;
307    }
308    /**
309     * Insert the method's description here.
310     * Creation date: (22-08-00 17:53:01)
311     * @return double
312     */
313    public final static int getANIMATION_MAX_SPEED() {
314        return ANIMATION_MAX_SPEED;
315    }
316    /**
317     * Insert the method's description here.
318     * Creation date: (22-08-00 17:56:11)

```

```

320     * @return double
321     */
322     public final static int getANIMATION_MIN_SPEED() {
323         return ANIMATION_MIN_SPEED;
324     }
325 /**
326     * Insert the method's description here.
327     * Creation date: (10-08-00 15:38:52)
328     * @return java.lang.String
329     */
330     public final static java.lang.String getATTEMPT_CONNECTION_CONFIRMATION_MESSAGE() {
331         return ATTEMPT_CONNECTION_CONFIRMATION_MESSAGE;
332     }
333 /**
334     * Insert the method's description here.
335     * Creation date: (17-07-00 13:52:34)
336     * @return java.lang.String
337     */
338     public final static java.lang.String getAUTHOR_NAME() {
339         return AUTHOR_NAME;
340     }
341 /**
342     * Insert the method's description here.
343     * Creation date: (14-10-00 11:31:51)
344     * @return java.lang.String
345     */
346     public final static java.lang.String getAUTOLOCK_NODES() {
347         return AUTOLOCK_NODES;
348     }
349 /**
350     * Insert the method's description here.
351     * Creation date: (16-06-00 17:55:28)
352     * @return int
353     */
354     public final static Color getBACKGROUND_COLOUR() {
355         return BACKGROUND_COLOUR;
356     }
357 /**
358     * Insert the method's description here.
359     * Creation date: (22-06-00 20:22:01)
360     * @return java.lang.String
361     */
362     public final static java.lang.String getBITMAP() {
363         return BITMAP;
364     }
365 /**
366     * Insert the method's description here.
367     * Creation date: (26-07-00 14:29:56)
368     * @return java.lang.String
369     */
370     public final static java.lang.String getBROADCASTING_DOCUMENT_REMOVALS() {
371         return BROADCASTING_DOCUMENT_REMOVALS;
372     }
373 /**
374     * Insert the method's description here.
375     * Creation date: (08-07-00 19:00:37)
376     * @return java.lang.String
377     * @param number0fDocuments int
378     * @param number0fRecipients int
379     */
380     public static String getBROADCASTING_DOCUMENTS_MESSAGE(int number0fDocuments) {
381         return "Broadcasting" + String.valueOf( number0fDocuments ) + " document(s)";
382     }
383 /**
384     * Insert the method's description here.
385     * Creation date: (08-07-00 19:00:37)

```

```

    * @return java.lang.String
386    * @param number_of_documents int
    * @param number_of_recipients int
388   */
public static String getBROADCASTING_DOCUMENTS_MESSAGE(int number_of_documents, int number_of_recipients) {
390   return "Broadcasting to other group members: " + String.valueOf( number_of_documents )+ " document(s) to " + String.valueOf( number_of_recipients )+ " 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 work_group peerviewmisc.WorkGroup
   */
414 public static String getClient_joined_group_message(peerviewmisc.WorkGroup workGroup) {
   return "You have now joined " + workGroup.getGroupName() + ".";
}
416 /**
   * Insert the method's description here.
   * Creation date: (21-08-00 08:39:23)
420   * @return java.lang.String
   * @param work_group 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 clients_author_name 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)

```

```

448     * @return java.lang.String
449     */
450     public final static java.lang.String getCONFIRM_CONNECT_TO_SERVER() {
451         return CONFIRM_CONNECT_TO_SERVER;
452     }
453 /**
454     * Insert the method's description here.
455     * Creation date: (12-07-00 11:09:48)
456     * @return java.lang.String
457     * @param messageID java.lang.String
458     */
459     public static String getCONFIRM_MESSAGE_DELETION(String messageID)
460     {
461         return "Delete\u00b7" + messageID + "\u00b7?";
462     }
463 /**
464     * Insert the method's description here.
465     * Creation date: (26-07-00 20:41:52)
466     * @param groupName java.lang.String
467     */
468     public static String getCONFIRM_WORKGROUP_DELETION(String groupName)
469     {
470         return "Delete\u00b7the\u00b7group:\u00b7" + "\u00b7" + groupName + "\u00b7?";
471     }
472 /**
473     * Insert the method's description here.
474     * Creation date: (10-08-00 20:39:43)
475     * @return java.lang.String
476     */
477     public final static java.lang.String getCONNECTED_TO_GROUP_MANAGEMENT() {
478         return CONNECTED_TO_GROUP_MANAGEMENT;
479     }
480 /**
481     * Insert the method's description here.
482     * Creation date: (09-10-00 12:57:14)
483     * @return java.lang.String
484     * @param connectionEventString java.lang.String
485     */
486     public static final String getConnection_FAILURE(String connectionEventString)
487     {
488         return "Connection\u00b7failure\u00b7communication\u00b7with\u00b7the\u00b7server\u00b7has\u00b7been\u00b7disrupted\u00b7(" + connectionEventString + ")";
489     }
490 /**
491     * Insert the method's description here.
492     * Creation date: (08-08-00 10:28:46)
493     * @return java.lang.String
494     */
495     public final static java.lang.String getCONTENTS_AND_SIGNATURE_DIVIDER() {
496         return CONTENTS_AND_SIGNATURE_DIVIDER;
497     }
498 /**
499     * Insert the method's description here.
500     * Creation date: (07-08-00 17:36:30)
501     * @return java.lang.String
502     */
503     public final static java.lang.String getCOPIED_TEXT_MESSAGE() {
504         return COPIED_TEXT_MESSAGE;
505     }
506 /**
507     * Insert the method's description here.
508     * Creation date: (10-08-00 02:27:09)
509     * @return java.lang.String
510     * @param groupName java.lang.String
511     */
512     public static String getCOULD_NOT_CONNECT_TO_GROUP_MESSAGE(String groupName)

```

```

512 {
513     return "Could not establish a connection to the group " + groupName + "\"";
514 }
515 /**
516 * Insert the method's description here.
517 * Creation date: (15-11-00 19:46:55)
518 * @return java.lang.String
519 */
520 public final static java.lang.String getCOULD_NOT_INITIALIZE_HELP() {
521     return COULD_NOT_INITIALIZE_HELP;
522 }
523 /**
524 * Insert the method's description here.
525 * Creation date: (06-09-00 21:45:12)
526 * @return java.lang.String
527 */
528 public final static java.lang.String getCOULD_NOT_READ_DOCUMENT_LIST() {
529     return COULD_NOT_READ_DOCUMENT_LIST;
530 }
531 /**
532 * Insert the method's description here.
533 * Creation date: (06-09-00 22:04:19)
534 * @return java.lang.String
535 */
536 public final static java.lang.String getCOULD_NOT_WRITE_DOCUMENT_LIST() {
537     return COULD_NOT_WRITE_DOCUMENT_LIST;
538 }
539 /**
540 * Insert the method's description here.
541 * Creation date: (19-07-00 10:48:58)
542 * @return java.lang.String
543 */
544 public final static java.lang.String getDEFAULT_DOCUMENT_HEIGHT() {
545     return DEFAULT_DOCUMENT_HEIGHT;
546 }
547 /**
548 * Insert the method's description here.
549 * Creation date: (19-07-00 10:47:47)
550 * @return java.lang.String
551 */
552 public final static java.lang.String getDEFAULT_DOCUMENT_WIDTH() {
553     return DEFAULT_DOCUMENT_WIDTH;
554 }
555 /**
556 * Insert the method's description here.
557 * Creation date: (03-08-00 17:21:36)
558 * @return java.lang.String
559 */
560 public final static java.lang.String getDEFAULT_FRAME_HORIZONTAL_SIZE() {
561     return DEFAULT_FRAME_HORIZONTAL_SIZE;
562 }
563 /**
564 * Insert the method's description here.
565 * Creation date: (03-08-00 17:21:54)
566 * @return java.lang.String
567 */
568 public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
569     return DEFAULT_FRAME_VERTICAL_SIZE;
570 }
571 /**
572 * Insert the method's description here.
573 * Creation date: (03-08-00 18:10:49)
574 * @return java.lang.String
575 */
576 public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
577     return DEFAULT_FRAME_X_COORDINATE;

```

```

578 }
579 /**
580 * Insert the method's description here.
581 * Creation date: (03-08-00 18:11:11)
582 * @return java.lang.String
583 */
584 public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
585     return DEFAULT_FRAME_Y_COORDINATE;
586 }
587 /**
588 * Insert the method's description here.
589 * Creation date: (06-09-00 00:48:38)
590 * @return java.lang.String
591 */
592 public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT()
593 {
594     return DEFAULT_GRID_LAYOUT_DOCUMENT_HEIGHT;
595 }
596 /**
597 * Insert the method's description here.
598 * Creation date: (06-09-00 00:47:52)
599 * @return java.lang.String
600 */
601 public final static java.lang.String getDEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH() {
602     return DEFAULT_GRID_LAYOUT_DOCUMENT_WIDTH;
603 }
604 /**
605 * Insert the method's description here.
606 * Creation date: (06-09-00 00:50:09)
607 * @return java.lang.String
608 */
609 public final static java.lang.String getDEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING() {
610     return DEFAULT_GRID_LAYOUT_HORIZONTAL_SPACING;
611 }
612 /**
613 * Insert the method's description here.
614 * Creation date: (06-09-00 00:50:42)
615 * @return java.lang.String
616 */
617 public final static java.lang.String getDEFAULT_GRID_LAYOUT_VERTICAL_SPACING() {
618     return DEFAULT_GRID_LAYOUT_VERTICAL_SPACING;
619 }
620 /**
621 * Insert the method's description here.
622 * Creation date: (11-08-00 17:12:57)
623 * @return java.lang.String
624 */
625 public final static java.lang.String getDEFAULT_GROUP_ID() {
626     return DEFAULT_GROUP_ID;
627 }
628 /**
629 * Insert the method's description here.
630 * Creation date: (19-07-00 10:52:34)
631 * @return java.lang.String
632 */
633 public final static java.lang.String getDEFAULT_HORIZONTAL_DOCUMENT_SPACING() {
634     return DEFAULT_HORIZONTAL_DOCUMENT_SPACING;
635 }
636 /**
637 * Insert the method's description here.
638 * Creation date: (28-06-00 13:37:45)
639 * @return double
640 */
641 public final static double getDEFAULT_PANORAMA_PERCENTUAL_SIZE() {
642     return DEFAULT_PANORAMA_PERCENTUAL_SIZE;
643 }

```

```

644 /**
645  * Insert the method's description here.
646  * Creation date: (25-06-00 12:17:08)
647  * @return java.lang.String[][]
648 */
649 public final static Properties getDefault_PREFERENCES()
650 {
651     Properties p = new Properties();
652     for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
653     {
654         p.setProperty( DEFAULT_PREFERENCES [i][0], DEFAULT_PREFERENCES [i][1] );
655     }
656     return p;
657 }
658 /**
659  * Insert the method's description here.
660  * Creation date: (19-07-00 10:53:28)
661  * @return java.lang.String
662 */
663 public final static java.lang.String getDefault_VERTICAL_DOCUMENT_SPACING() {
664     return DEFAULT_VERTICAL_DOCUMENT_SPACING;
665 }
666 /**
667  * Insert the method's description here.
668  * Creation date: (25-06-00 20:55:23)
669  * @return java.lang.String[][]
670 */
671 public final static Hashtable getDISPLAY_QUALITIES()
672 {
673     Hashtable displayQualities = new Hashtable();
674     for ( int i = 0 ; i < DISPLAY_QUALITIES.length; i++ )
675     {
676         displayQualities.put( DISPLAY_QUALITIES [i][0], DISPLAY_QUALITIES [i][1] );
677     }
678     return displayQualities;
679 }
680 /**
681  * Insert the method's description here.
682  * Creation date: (25-06-00 16:36:00)
683  * @return java.lang.String
684 */
685 public final static java.lang.String getDISPLAY_QUALITY() {
686     return DISPLAY_QUALITY;
687 }
688 /**
689  * Insert the method's description here.
690  * Creation date: (08-07-00 13:59:10)
691  * @return java.lang.String
692 */
693 public final static java.lang.String getDISTRIBUTED_DOCUMENT() {
694     return DISTRIBUTED_DOCUMENT;
695 }
696 /**
697  * Insert the method's description here.
698  * Creation date: (15-08-00 17:50:16)
699  * @return java.lang.String
700 */
701 public final static java.lang.String getDO_NOT_ZOOM_TO_OVERVIEW() {
702     return DO_NOT_ZOOM_TO_OVERVIEW;
703 }
704 /**
705  * Insert the method's description here.
706  * Creation date: (30-07-00 20:40:10)
707  * @return int
708 */

```

```

710 public final static int getDOCUMENT_BORDER_WIDTH() {
711     return DOCUMENT_BORDER_WIDTH;
712 }
713 /**
714 * Insert the method's description here.
715 * Creation date: (22-08-00 01:26:41)
716 * @return java.lang.String
717 * @param documentName java.lang.String
718 * @param authorName java.lang.String
719 * @param size long
720 * @param lastUpdate java.util.Date
721 */
722 public final static String getDOCUMENT_LABEL(String documentName, String authorName, long size, java.
723     util.Date lastUpdate)
724 { java.util.GregorianCalendar calendar = new java.util.GregorianCalendar();
725     calendar.setTime( lastUpdate );
726     return "Name:" + documentName + "\n" +
727         "Author:" + authorName + "\n" +
728         "Size:" + String.valueOf( size )+ "bytes" + "\n" +
729         "LastUpdated:" + (new java.text.SimpleDateFormat()).format( lastUpdate )+ ":" + String.valueOf(
730             calendar.get( java.util.Calendar.SECOND ) );
731 }
732 /**
733 * Insert the method's description here.
734 * Creation date: (30-07-00 20:49:44)
735 * @return java.awt.Color
736 */
737 public final static java.awt.Color getDOCUMENT_LINE_BORDER_COLOUR() {
738     return DOCUMENT_LINE_BORDER_COLOUR;
739 }
740 /**
741 * Insert the method's description here.
742 * Creation date: (06-09-00 21:36:21)
743 * @return java.lang.String
744 */
745 public final static java.lang.String getDOCUMENT_LIST_FILENAME()
746 {
747     return DOCUMENT_LIST_FILENAME;
748 }
749 /**
750 * Insert the method's description here.
751 * Creation date: (30-07-00 20:44:17)
752 * @return java.awt.Color
753 */
754 public final static java.awt.Color getDOCUMENT_MATTE_BORDER_COLOUR() {
755     return DOCUMENT_MATTE_BORDER_COLOUR;
756 }
757 /**
758 * Insert the method's description here.
759 * Creation date: (21-08-00 12:34:32)
760 * @return java.lang.String
761 */
762 public final static java.lang.String getDOCUMENT_NAME() {
763     return DOCUMENT_NAME;
764 }
765 /**
766 * Insert the method's description here.
767 * Creation date: (09-07-00 17:44:32)
768 * @return java.lang.String
769 */
770 public final static java.lang.String getDOCUMENT_PATH() {
771     return DOCUMENT_PATH;
772 }
773 /**
774 * Insert the method's description here.
775 * Creation date: (06-09-00 14:52:54)

```

```

774 * @return java.lang.String
775 */
776 public static final String getDOCUMENT_SIZE_OUT_OF_BOUNDS()
{
777     return "Documents\u00b7must\u00b7be\u00b7between\u00b7" + String.valueOf( getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() )+ "\u00b7and\u00b7"
778         + String.valueOf( getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() )+ "\u00b7in\u00b7width\u00b7and\u00b7between\u00b7"
779         + String.valueOf( getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() )+ "\u00b7and\u00b7"
780         + String.valueOf( getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() )+ "\u00b7in\u00b7height.\u00b7";
781 }
782 /**
783 * Insert the method's description here.
784 * Creation date: (21-08-00 16:54:07)
785 * @return java.lang.String
786 */
787 public final static java.lang.String getDOCUMENT_UPDATE() {
788     return DOCUMENT_UPDATE;
789 }
790 /**
791 * Insert the method's description here.
792 * Creation date: (10-08-00 20:46:02)
793 * @return java.lang.String
794 */
795 public final static java.lang.String getFAILED_TO_CONNECT_TO_SERVER() {
796     return FAILED_TO_CONNECT_TO_SERVER;
797 }
798 /**
799 * Insert the method's description here.
800 * Creation date: (10-08-00 16:02:45)
801 * @return java.lang.String
802 */
803 public final static java.lang.String getFAILED_TO_RECEIVE_GROUP_DIRECTORY() {
804     return FAILED_TO_RECEIVE_GROUP_DIRECTORY;
805 }
806 /**
807 * Insert the method's description here.
808 * Creation date: (30-08-00 17:27:03)
809 * @return java.lang.String
810 */
811 public final static java.lang.String getFALSE() {
812     return FALSE;
813 }
814 /**
815 * Insert the method's description here.
816 * Creation date: (22-06-00 19:28:09)
817 * @return int
818 */
819 public final static int getFile_EXTENSION_INDEX() {
820     return FILE_EXTENSION_INDEX;
821 }
822 /**
823 * Insert the method's description here.
824 * Creation date: (22-06-00 19:28:32)
825 * @return int
826 */
827 public final static int getFile_TYPE_DESCRIPTION_INDEX() {
828     return FILE_TYPE_DESCRIPTION_INDEX;
829 }
830 /**
831 * Insert the method's description here.
832 * Creation date: (23-02-01 08:49:05)
833 * @return java.lang.String[][]
834 */
835 public final static java.lang.String[][] getFile_TYPE_SYNONYM_TABLE() {
836     return FILE_TYPE_SYNONYM_TABLE;
837 }
838 /**

```

```

840 * Insert the method's description here.
841 * Creation date: (23-02-01 08:42:21)
842 * @return java.util.Hashtable
843 */
844 public static java.util.Hashtable getFILE_TYPE_SYNONYMS()
{
845     if ( FILE_TYPE_SYNONYMS == null )
846     {
847         FILE_TYPE_SYNONYMS = new Hashtable();
848         for ( int i = 0; i < getFileTypeSynonymTable().length; i++ )
849         {
850             java.util.ArrayList synonymList = new java.util.ArrayList();
851             for (int j = 0; j < getFileTypeSynonymTable()[i].length; j++ )
852             {
853                 synonymList.add( getFileTypeSynonymTable()[i][j] );
854             }
855             FILE_TYPE_SYNONYMS.put( getFileTypeSynonymTable()[i][0], synonymList );
856         }
857     }
858     return FILE_TYPE_SYNONYMS;
859 }
860 /**
861 * Insert the method's description here.
862 * Creation date: (20-06-00 08:23:58)
863 * @return java.lang.String[]
864 */
865 public final static java.lang.String[][] getFileTypes() {
866     return FileTypes;
867 }
868 /**
869 * Insert the method's description here.
870 * Creation date: (30-08-00 22:01:04)
871 * @return java.lang.String
872 */
873 public final static java.lang.String getFINISHED_ADDING_DOCUMENTS() {
874     return FINISHED_ADDING_DOCUMENTS;
875 }
876 /**
877 * Insert the method's description here.
878 * Creation date: (03-08-00 17:20:38)
879 * @return java.lang.String
880 */
881 public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
882     return FRAME_HORIZONTAL_SIZE;
883 }
884 /**
885 * Insert the method's description here.
886 * Creation date: (05-08-00 20:11:48)
887 * @return double
888 */
889 public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
890     return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
891 }
892 /**
893 * Insert the method's description here.
894 * Creation date: (03-08-00 17:21:19)
895 * @return java.lang.String
896 */
897 public final static java.lang.String getFRAME_VERTICAL_SIZE() {
898     return FRAME_VERTICAL_SIZE;
899 }
900 /**
901 * Insert the method's description here.
902 * Creation date: (03-08-00 18:09:16)
903 * @return java.lang.String
904 */

```

```

906  /*
907   * Insert the method's description here.
908   * Creation date: (03-08-00 18:09:58)
909   * @return java.lang.String
910  */
911  public final static java.lang.String getFRAME_X_COORDINATE() {
912      return FRAME_X_COORDINATE;
913  }
914 /**
915  * Insert the method's description here.
916  * Creation date: (25-06-00 13:32:26)
917  * @return java.lang.String
918 */
919 public final static java.lang.String getFRAME_Y_COORDINATE() {
920     return FRAME_Y_COORDINATE;
921 }
922 /**
923  * Insert the method's description here.
924  * Creation date: (19-07-00 13:00:42)
925  * @return java.lang.String
926 */
927 public final static java.lang.String getGRID_LAYOUT() {
928     return GRID_LAYOUT;
929 }
930 /**
931  * Insert the method's description here.
932  * Creation date: (19-07-00 13:00:42)
933  * @return java.lang.String
934 */
935 public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() {
936     return GRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS;
937 }
938 /**
939  * Insert the method's description here.
940  * Creation date: (19-07-00 13:00:07)
941  * @return java.lang.String
942 */
943 public final static java.lang.String getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() {
944     return GRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS;
945 }
946 /**
947  * Insert the method's description here.
948  * Creation date: (06-09-00 15:05:25)
949  * @return java.lang.String
950 */
951 public final static String getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS()
952 {
953     return "Horizontal\u00b7document\u00b7spacing\u00b7must\u00b7be\u00b7between\u00b7" +
954         getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() + "\u00b7and\u00b7" +
955         getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() + "." +
956         "Vertical\u00b7document\u00b7spacing\u00b7must\u00b7be\u00b7between\u00b7" +
957         getMIN_GRID_LAYOUT_VERTICAL_SPACING() + "\u00b7and\u00b7" +
958         getMAX_GRID_LAYOUT_VERTICAL_SPACING() + ".";
959 }
960 /**
961  * Insert the method's description here.
962  * Creation date: (25-07-00 14:56:17)
963  * @return java.lang.String
964 */
965 public final static java.lang.String getGROUP_AT_MAXIMUM_MEMBERS_MESSAGE() {
966     return GROUP_AT_MAXIMUM_MEMBERS_MESSAGE;
967 }
968 /**
969  * Insert the method's description here.
970  * Creation date: (18-08-00 02:02:33)
971  * @return int
972 */
973 public final static String getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() {
974     return GROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS;
975 }

```

```

972 /**
 * Insert the method's description here.
974 * Creation date: (18-08-00 02:06:15)
 * @return java.lang.String
976 */
977 public final static java.lang.String getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() {
978     return GROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS;
979 }
980 /**
 * Insert the method's description here.
982 * Creation date: (21-07-00 12:48:25)
 * @return int
984 */
985 public final static int getGROUP_DIRECTORY_COLUMN_WIDTH() {
986     return GROUP_DIRECTORY_COLUMN_WIDTH;
987 }
988 /**
 * Insert the method's description here.
989 * Creation date: (10-08-00 15:39:59)
 * @return int
990 */
991 public final static int getGROUP_DIRECTORY_REQUEST_DELAY() {
992     return GROUP_DIRECTORY_REQUEST_DELAY;
993 }
994 /**
 * Insert the method's description here.
995 * Creation date: (26-07-00 20:32:52)
 * @return java.lang.String
996 */
997 public final static java.lang.String getGROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE() {
998     return GROUP_HAS_ACTIVE_PARTICIPANTS_ERROR_MESSAGE;
999 }
1000 /**
 * Insert the method's description here.
1001 * Creation date: (22-06-00 20:20:34)
 * @return java.lang.String
1002 */
1003 public final static java.lang.String getHTML() {
1004     return HTML;
1005 }
1006 /**
 * Insert the method's description here.
1007 * Creation date: (18-07-00 11:23:18)
 * @return java.lang.String
1008 */
1009 public final static java.lang.String getINVALID_DELETION_ATTEMPTED_MESSAGE() {
1010     return INVALID_DELETION_ATTEMPTED_MESSAGE;
1011 }
1012 /**
 * Insert the method's description here.
1013 * Creation date: (21-08-00 12:26:41)
 * @return java.lang.String
1014 */
1015 public final static java.lang.String getLast_MODIFIED() {
1016     return LAST_MODIFIED;
1017 }
1018 /**
 * Insert the method's description here.
1019 * Creation date: (25-06-00 16:38:21)
 * @return java.lang.String
1020 */
1021 public final static java.lang.String getLAYOUT_SCHEME() {
1022     return LAYOUT_SCHEME;
1023 }
1024 /**
 * Insert the method's description here.
1025 * Creation date: (21-07-00 12:48:25)
 * @return int
1026 */
1027 public final static int getLAYOUT_SCHEME_WIDTH() {
1028     return LAYOUT_SCHEME_WIDTH;
1029 }
1030 /**
 * Insert the method's description here.
1031 * Creation date: (21-07-00 12:48:25)
 * @return int
1032 */
1033 public final static int getLAYOUT_SCHEME_HEIGHT() {
1034     return LAYOUT_SCHEME_HEIGHT;
1035 }
1036 /**
 * Insert the method's description here.

```

```

1038 * Creation date: (25-06-00 21:32:43)
1039 * @return java.lang.String[]
1040 */
1041 public final static java.lang.String[] getLAYOUT_SCHEMES() {
1042     return LAYOUT_SCHEMES;
1043 }
1044 /**
1045 * Insert the method's description here.
1046 * Creation date: (06-09-00 20:36:54)
1047 * @return java.lang.String
1048 */
1049 public final static java.lang.String getLOAD_DOCUMENT_LIST_ON_STARTUP() {
1050     return LOAD_DOCUMENT_LIST_ON_STARTUP;
1051 }
1052 /**
1053 * Insert the method's description here.
1054 * Creation date: (07-07-00 23:09:39)
1055 * @return int
1056 */
1057 public final static int getLOCAL_DOCUMENT() {
1058     return LOCAL_DOCUMENT;
1059 }
1060 /**
1061 * Insert the method's description here.
1062 * Creation date: (26-06-00 09:55:24)
1063 * @return java.lang.String
1064 */
1065 public final static java.lang.String getLOCAL_HOST() {
1066     return LOCAL_HOST;
1067 }
1068 /**
1069 * Insert the method's description here.
1070 * Creation date: (10-08-00 04:40:26)
1071 * @return java.lang.String
1072 */
1073 public final static java.lang.String getLOCAL_MODE_MESSAGE() {
1074     return LOCAL_MODE_MESSAGE;
1075 }
1076 /**
1077 * Insert the method's description here.
1078 * Creation date: (10-10-00 00:09:18)
1079 * @return int
1080 */
1081 public final static int getMax_DOCNAME_LENGTH() {
1082     return MAX_DOCNAME_LENGTH;
1083 }
1084 /**
1085 * Insert the method's description here.
1086 * Creation date: (06-09-00 14:48:29)
1087 * @return int
1088 */
1089 public final static int getMax_GRID_LAYOUT_DOCUMENT_HEIGHT() {
1090     return MAX_GRID_LAYOUT_DOCUMENT_HEIGHT;
1091 }
1092 /**
1093 * Insert the method's description here.
1094 * Creation date: (06-09-00 14:50:48)
1095 * @return int
1096 */
1097 public final static int getMax_GRID_LAYOUT_DOCUMENT_WIDTH() {
1098     return MAX_GRID_LAYOUT_DOCUMENT_WIDTH;
1099 }
1100 /**
1101 * Insert the method's description here.
1102 * Creation date: (06-09-00 14:58:01)
1103 * @return int

```

```

1104  /*
1105   * Insert the method's description here.
1106   * Creation date: (06-09-00 15:02:56)
1107   * @return int
1108  */
1109  public final static int getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() {
1110      return MAX_GRID_LAYOUT_HORIZONTAL_SPACING;
1111  }
1112 /**
1113  * Insert the method's description here.
1114  * Creation date: (27-07-00 12:35:59)
1115  * @return java.util.Hashtable
1116  */
1117  public final static java.util.Hashtable getMEDIA_TYPES()
1118  {
1119      if ( MEDIA_TYPES.size() == 0 )
1120      {
1121          for ( int i = 0; i < getFileTypes().length; i++ )
1122          {
1123              MEDIA_TYPES.put( getFileTypes()[i][0], getFileTypes()[i][2] );
1124          }
1125      }
1126      return MEDIA_TYPES;
1127  }
1128 /**
1129  * Insert the method's description here.
1130  * Creation date: (18-06-00 17:33:32)
1131  * @return java.lang.String
1132  */
1133  public final static java.lang.String getMessage_AREA_DEFAULT_STRING() {
1134      return MESSAGE_AREA_DEFAULT_STRING;
1135  }
1136 /**
1137  * Insert the method's description here.
1138  * Creation date: (29-10-00 16:42:19)
1139  * @return java.lang.String
1140  */
1141  public final static java.lang.String getMessage_LOG_FILENAME() {
1142      return MESSAGE_LOG_FILENAME;
1143  }
1144 /**
1145  * Insert the method's description here.
1146  * Creation date: (29-10-00 16:54:58)
1147  * @return java.lang.String
1148  */
1149  public final static java.lang.String getMessage_LOG_MESSAGE() {
1150      return MESSAGE_LOG_MESSAGE;
1151  }
1152 /**
1153  * Insert the method's description here.
1154  * Creation date: (29-10-00 16:43:20)
1155  * @return java.lang.String
1156  */
1157  public final static java.lang.String getMessage_LOG_WRITE_ERROR() {
1158      return MESSAGE_LOG_WRITE_ERROR;
1159  }
1160 /**
1161  * Insert the method's description here.
1162  * Creation date: (19-07-00 20:06:48)
1163  * @return java.lang.String
1164  */
1165  public final static java.lang.String getMessage_PROPERTIES_NAME_COLUMN_HEADING() {

```

```

1170     return MESSAGE_PROPERTIES_NAME_COLUMN_HEADING;
1171 }
1172 /**
1173 * Insert the method's description here.
1174 * Creation date: (19-07-00 20:07:10)
1175 * @return java.lang.String
1176 */
1177 public final static java.lang.String getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADING() {
1178     return MESSAGE_PROPERTIES_VALUE_COLUMN_HEADING;
1179 }
1180 /**
1181 * Insert the method's description here.
1182 * Creation date: (06-09-00 14:49:17)
1183 * @return int
1184 */
1185 public final static int getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() {
1186     return MIN_GRID_LAYOUT_DOCUMENT_HEIGHT;
1187 }
1188 /**
1189 * Insert the method's description here.
1190 * Creation date: (06-09-00 14:51:49)
1191 * @return int
1192 */
1193 public final static int getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() {
1194     return MIN_GRID_LAYOUT_DOCUMENT_WIDTH;
1195 }
1196 /**
1197 * Insert the method's description here.
1198 * Creation date: (06-09-00 15:00:54)
1199 * @return int
1200 */
1201 public final static int getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() {
1202     return MIN_GRID_LAYOUT_HORIZONTAL_SPACING;
1203 }
1204 /**
1205 * Insert the method's description here.
1206 * Creation date: (06-09-00 15:02:09)
1207 * @return int
1208 */
1209 public final static int getMIN_GRID_LAYOUT_VERTICAL_SPACING() {
1210     return MIN_GRID_LAYOUT_VERTICAL_SPACING;
1211 }
1212 /**
1213 * Insert the method's description here.
1214 * Creation date: (25-06-00 16:33:43)
1215 * @return java.lang.String
1216 */
1217 public final static java.lang.String getName() {
1218     return NAME;
1219 }
1220 /**
1221 * Insert the method's description here.
1222 * Creation date: (13-08-00 19:55:27)
1223 * @return java.lang.String
1224 */
1225 public final static java.lang.String getNO_GROUP_SELECTED_TO_DELETE() {
1226     return NO_GROUP_SELECTED_TO_DELETE;
1227 }
1228 /**
1229 * Insert the method's description here.
1230 * Creation date: (26-07-00 18:23:40)
1231 * @return java.lang.String
1232 */
1233 public final static java.lang.String getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() {
1234     return NO_GROUP_SELECTED_TO_EDIT_MESSAGE;
1235 }

```

```

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)
1303 * @return java.lang.String
1304 */
1305 public final static java.lang.String getPANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE() {
1306     return PANORAMA_ALREADY_CONTAINED_DOCUMENTS_MESSAGE;
1307 }
1308 /**
1309 * Insert the method's description here.
1310 * Creation date: (22-06-00 20:20:00)
1311 * @return java.lang.String
1312 */
1313 public final static java.lang.String getPLAIN_TEXT() {
1314     return PLAIN_TEXT;
1315 }
1316 /**
1317 * Insert the method's description here.
1318 * Creation date: (25-06-00 19:29:14)
1319 * @return java.lang.String
1320 */
1321 public final static java.lang.String getPREFERENCES_FILE_NAME() {
1322     return PREFERENCES_FILE_NAME;
1323 }
1324 /**
1325 * Insert the method's description here.
1326 * Creation date: (17-06-00 19:40:40)
1327 * @return java.lang.String
1328 */
1329 public final static java.lang.String getPROCESSING_DOCUMENTS() {
1330     return PROCESSING_DOCUMENTS;
1331 }
1332 /**
1333 * Insert the method's description here.
1334 * Creation date: (26-06-00 08:13:08)
1335 * @return java.lang.String
1336 */
1337 public final static java.lang.String getPROPERTIES_HEADER() {
1338     return PROPERTIES_HEADER;
1339 }
1340 /**
1341 * Insert the method's description here.
1342 * Creation date: (21-08-00 15:49:38)
1343 * @return java.lang.String
1344 */
1345 public final static java.lang.String getRAW_BITMAP() {
1346     return RAW_BITMAP;
1347 }
1348 /**
1349 * Insert the method's description here.
1350 * Creation date: (19-08-00 21:35:31)
1351 * @param documentName java.lang.String
1352 * @param senderName java.lang.String
1353 */
1354 public final static String getREMOVING_DISTRIBUTED_DOCUMENT(String documentName, String senderName)
1355 {
1356     return senderName + " has requested that " + documentName + " be removed from the panorama.";
1357 }
1358 /**
1359 * Insert the method's description here.
1360 * Creation date: (05-09-00 15:15:12)
1361 * @param authorName java.lang.String
1362 * @param numDoks int
1363 */
1364 public final static String getREMOVING_DISTRIBUTED_DOCUMENTS_BY_REQUEST(String authorName, int numDoks)
1365 {
1366     return "Removing " + numDoks + " documents from the panorama by request from " + authorName;
1367 }

```

```

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 number0fDocuments String
1404 */
public static String getRequestING_DOCUMENTS_MESSAGE(int number0fDocuments)
1406 {
    return "Receiving\u0026adding\u0026total\u0026of\u0026" + String.valueOf( number0fDocuments )+ "\u0026documents\u0026from\u0026the
           \u0026other\u0026members\u0026of\u0026the\u0026group." ;
}
1408 /**
 * 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.
1435     * Creation date: (19-06-00 22:01:04)
1436     * @return java.lang.String
1437     */
1438     public final static java.lang.String getRETRIEVING_DOCUMENTS() {
1439         return RETRIEVING_DOCUMENTS;
1440     }
1441     /**
1442     * Insert the method's description here.
1443     * Creation date: (22-06-00 20:21:32)
1444     * @return java.lang.String
1445     */
1446     public final static java.lang.String getRTF() {
1447         return RTF;
1448     }
1449     /**
1450     * Insert the method's description here.
1451     * Creation date: (14-07-00 14:38:06)
1452     * @return double
1453     */
1454     public final static double getSCALE_FACTOR_SCALING() {
1455         return SCALE_FACTOR_SCALING;
1456     }
1457     /**
1458     * Insert the method's description here.
1459     * Creation date: (30-08-00 16:45:24)
1460     * @return java.lang.String
1461     */
1462     public final static java.lang.String getSENDER_NAME() {
1463         return SENDER_NAME;
1464     }
1465     /**
1466     * Insert the method's description here.
1467     * Creation date: (30-08-00 17:24:57)
1468     * @return java.lang.String
1469     */
1470     public final static java.lang.String getSHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX() {
1471         return SHOW_FULL_PATHNAMES_IN_VISIBLE_DOCUMENTS_BOX;
1472     }
1473     /**
1474     * Insert the method's description here.
1475     * Creation date: (25-06-00 16:35:18)
1476     * @return java.lang.String
1477     */
1478     public final static java.lang.String getSIGNATURE() {
1479         return SIGNATURE;
1480     }
1481     /**
1482     * Insert the method's description here.
1483     * Creation date: (05-10-00 23:11:42)
1484     * @return int
1485     */
1486     public final static int getTEXT_DOCUMENT_THRESHOLD() {
1487         return TEXT_DOCUMENT_THRESHOLD;
1488     }
1489     /**
1490     * Insert the method's description here.
1491     * Creation date: (11-08-00 00:22:49)
1492     * @return java.lang.String
1493     */
1494     public final static java.lang.String getTitle_REQUIRED_MESSAGE() {
1495         return TITLE_REQUIRED_MESSAGE;
1496     }
1497     /**
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 number_of_documents int
1518   * @param recipientName java.lang.String
    */
1520 public static String getUNICASTING_DOCUMENTS_MESSAGE(int number_of_documents, String recipientName) {
    return "New_group_member: unicasting" + String.valueOf( number_of_documents ) + " 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.
1565 * Creation date: (21-08-00 23:11:12)
1566 * @return java.lang.String
1567 * @param documentName java.lang.String
1568 * @param senderName java.lang.String
1569 */
1570 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName) {
1571     return "Received\u00a9update\u00a9of\u00a9" + documentName + ".\u00a9Now\u00a9updating...";
1572 }
1573 /**
1574 * Insert the method's description here.
1575 * Creation date: (21-08-00 23:11:12)
1576 * @return java.lang.String
1577 * @param documentName java.lang.String
1578 * @param senderName java.lang.String
1579 */
1580 public final static String getUPDATING_DISTRIBUTED_DOCUMENT(String documentName, String senderName) {
1581     return "Received\u00a9update\u00a9of\u00a9" + documentName + "\u00a9from\u00a9" + senderName + ".\u00a9Now\u00a9updating...";
1582 }
1583 /**
1584 * Insert the method's description here.
1585 * Creation date: (21-08-00 20:42:58)
1586 * @return java.lang.String
1587 */
1588 public final static java.lang.String getUPDATING_DOCUMENTS() {
1589     return UPDATING_DOCUMENTS;
1590 }
1591 /**
1592 * Insert the method's description here.
1593 * Creation date: (30-08-00 10:43:08)
1594 * @return java.lang.String
1595 */
1596 public final static java.lang.String getVISIBLE_DOCUMENTS_BOX_CAPTION() {
1597     return VISIBLE_DOCUMENTS_BOX_CAPTION;
1598 }
1599 /**
1600 * Insert the method's description here.
1601 * Creation date: (30-08-00 14:17:20)
1602 * @return int
1603 */
1604 public final static int getVISIBLE_DOCUMENTS_ICON_TEXT_GAP() {
1605     return VISIBLE_DOCUMENTS_ICON_TEXT_GAP;
1606 }
1607 /**
1608 * Insert the method's description here.
1609 * Creation date: (01-09-00 07:50:17)
1610 * @return java.awt.Dimension
1611 */
1612 public final static java.awt.Dimension getVISIBLE_DOCUMENTS_ITEM_MAX_SIZE() {
1613     return VISIBLE_DOCUMENTS_ITEM_MAX_SIZE;
1614 }
1615 /**
1616 * Insert the method's description here.
1617 * Creation date: (31-08-00 00:15:59)
1618 * @return java.awt.Color
1619 */
1620 public final static java.awt.Color getVISIBLE_DOCUMENTS_SELECTION_COLOUR() {
1621     return VISIBLE_DOCUMENTS_SELECTION_COLOUR;
1622 }
1623 /**
1624 * Insert the method's description here.
1625 * Creation date: (30-08-00 14:26:51)
1626 * @return java.lang.String
1627 */
1628 public final static java.lang.String getVISIBLE_DOCUMENTS_TAB_SIZE() {
1629     return VISIBLE_DOCUMENTS_TAB_SIZE;

```

```

1630 }
1631 /**
1632 * Insert the method's description here.
1633 * Creation date: (15-08-00 17:49:18)
1634 * @return java.lang.String
1635 */
1636 public final static java.lang.String getZOOM_TO_OVERVIEW() {
1637     return ZOOM_TO_OVERVIEW;
1638 }
1639 /**
1640 * Insert the method's description here.
1641 * Creation date: (15-08-00 17:40:30)
1642 * @return java.lang.String
1643 */
1644 public final static java.lang.String getZOOM_TO_OVERVIEW_ON_UPDATE() {
1645     return ZOOM_TO_OVERVIEW_ON_UPDATE;
1646 }
1647 /**
1648 * Insert the method's description here.
1649 * Creation date: (30-07-00 19:10:52)
1650 * @return java.awt.Color
1651 */
1652 public final static java.awt.Color getZOOM_VECTOR_COLOUR() {
1653     return ZOOM_VECTOR_COLOUR;
1654 }
1655 /**
1656 * Insert the method's description here.
1657 * Creation date: (31-07-00 18:02:18)
1658 * @return int
1659 */
1660 public final static int getZOOM_VECTOR_WIDTH() {
1661     return ZOOM_VECTOR_WIDTH;
1662 }
1663 /**
1664 * Insert the method's description here.
1665 * Creation date: (22-06-00 19:51:35)
1666 */
1667 public void initialize()
1668 {
1669 }
1670 }
1671 /**
1672 * Insert the method's description here.
1673 * Creation date: (23-02-01 08:42:21)
1674 * @param newFILE_TYPE_SYNONYMS java.util.Hashtable
1675 */
1676 public static void setFILE_TYPE_SYNONYMS(java.util.Hashtable newFILE_TYPE_SYNONYMS) {
1677     FILE_TYPE_SYNONYMS = newFILE_TYPE_SYNONYMS;
1678 }
1679 }
```

Listing C.7: ClientPanorama.java

```

package clientapp;
2
3 /**
4 * This class represents the client panorama
5 * Creation date: (15-06-00 00:26:38)
6 * @author:
7 */
8 import java.awt.Color;
9 import java.awt.Dimension;
10 import java.awt.event.*;
11 import java.lang.reflect.InvocationTargetException;
12 import java.util.Hashtable;
13 import java.io.*;
14 import java.util.Enumeration;
```

```

import java.util.ArrayList;
16
import javax.swing.border.*;
18 import javax.swing.*;
import javax.swing.JProgressBar.*;
20 import javax.swing.Timer.*;
import javax.swing.SwingUtilities.*;
22
import edu.umd.cs.jazz.*;
24 import edu.umd.cs.jazz.util.*;
import edu.umd.cs.jazz.event.*;
26 import edu.umd.cs.jazz.component.*;
import java.awt.geom.*;
28
import java.awt.Point;
30 import javax.swing.event.MenuDragMouseListener;
import javax.swing.event.MenuDragMouseEvent;
32 import java.util.Collection;
import java.util.Iterator;
34
/** This class implements the panorama that occupies the center portion of the client application window
36 */
38 public class ClientPanorama extends ZCanvas
{
    /** Event handler to ensure panorama updates when documents are added or removed
40 */
42     public class UpdateHandler extends ZGroupAdapter
    {
        public void nodeAdded( ZGroupEvent zge )
44     {
            if ( zge.getChild() instanceof ZVisualLeaf )
46         {
            validateLayout();
48         }
50     }
52     public void nodeRemoved( ZGroupEvent zge )
    {
54         if ( zge.getChild() instanceof ZVisualLeaf )
56         {
            validateLayout();
58         }
59     }
59 };
60
/** Event handler triggered by a timer to ensure continuous updating of the local documents at user
determined intervals
62 */
64     public class UpdateActionListener implements ActionListener
66     {
        public void actionPerformed( ActionEvent ae )
68     {
            if ( isPanoramaLock() == false )
70         {
            getUpdateTimer().stop();
72             updateDocuments();
            getUpdateTimer().restart();
74         }
75     }
76
public class DocumentMouseListener extends ZMouseAdapter
77     {
78         public void mouseClicked( ZMouseEvent zme )
79         {
80             zme.consume();
81         }
82     }
83

```

```

80     };
82
83     public class CameraListener implements ZCameraListener
84     {
85         public void viewChanged( ZCameraEvent zce )
86         {
87             ...
88         }
89
90     /** This listener handles user clicks on documents in the panorama.
91      */
92     public class CameraMouseListener extends ZMouseAdapter
93     {
94         public void mouseClicked ( ZMouseEvent zme )
95         {
96             // if double click
97             if ( zme.getClickCount() == 2 )
98             {
99                 owner.getClientManager().clearTopicTree();
100                zoomToOverview();
101            }
102            if ( isResizingNode() )
103            {
104                doResize( zme );
105            }
106            if ( isMovingNode() )
107            {
108                // previousMousePosition = null;
109                setMovingNode( false );
110            }
111            zme.consume();
112            popDownPopUpMenu();
113        }
114    }
115
116    /** This listener handles clicks on the panorama canvas, i.e. outside any documents.
117     */
118    public class PanoramaMouseListener extends ZMouseAdapter
119    {
120        public void mouseClicked( ZMouseEvent zme )
121        {
122            // if double click
123            if ( zme.getClickCount() == 2 )
124            {
125                centerDocument( (ZVisualLeaf) zme.getNode() );
126                zme.consume();
127            }
128        }
129
130        public void mousePressed( ZMouseEvent zme )
131        {
132            if ( zme.getSource() instanceof ZVisualLeaf )
133            {
134                // System.out.println(" Fyr ");
135                ( (ZSwing) ( (ZVisualLeaf) zme.getSource() ).getVisualComponent() ).getComponent().
136                dispatchEvent( zme );
137            }
138
139            if ( isResizingNode() )
140            {
141                doResize( zme );
142            }
143            if ( isMovingNode() )
144            {
145                // lock node to current position if auto-lock is enabled

```

```

        getCurrentlyClickedNode().putClientProperty( ClientConstants.getNODE_LOCKED(), ClientInfo.
            getPreferences().getProperty( ClientConstants.getAUTOLOCK_NODES() ) );
146    // previousMousePosition = null;
147    setMovingNode( false );
148 }
149 // if right click, bring up the document pop-up menu
150 if ( zme.isMetaDown() )
151 {
152     popUpPopupMenu( (ZVisualLeaf) zme.getNode(), zme.getPoint() );
153     zme.consume();
154 }
155 else
156 {
157     popDownPopupMenu();
158 }
159 }

160 public void mouseReleased( ZMouseEvent zme )
161 {
162 }
163 }

164 }

165 public class PanoramaMouseMotionListener extends ZMouseMotionAdapter
166 {
167     public void mouseMoved( ZMouseEvent zme )
168     {
169         // a cleaner solution would be to use the mouseExited event in a mouselistener on the popup menu
170         // but that
171         // seemed to fail due to improper generation of motion events
172         // popDownPopupMenu();
173         if ( isResizingNode() == true )
174         {
175             ZBounds bounds = currentlyClickedNode.getVisualComponentBounds();
176             Point mousePosition = new Point( zme.getX(), zme.getY() );
177             Point upperLeftCorner = new Point( (int)bounds.getMinX(), (int)bounds.getMinY() );
178             // convert to global coordinates
179             getCamera().cameraToLocal( mousePosition, currentlyClickedNode );
180             //getCamera().cameraToLocal( upperLeftCorner, null );
181             Dimension newSize = new Dimension();
182             newSize.setSize(
183                 (int) mousePosition.getX() - (int) upperLeftCorner.getX(),
184                 (int) mousePosition.getY() - (int) upperLeftCorner.getY() );
185             ((PZSwing)currentlyClickedNode.getVisualComponent() ).getComponent().setPreferredSize(
186                 newSize );
187             ((PZSwing)currentlyClickedNode.getVisualComponent() ).computeBounds();
188             currentlyClickedNode.reshape();
189             revalidate();
190             getDrawingSurface().repaint();
191         }
192         if ( isMovingNode() == true )
193         {
194             // I tried a variety of approaches to implementing the below translation code for panorama
195             // nodes, i.e. documents.
196             // These included calculating the difference between the position between the most current
197             // mouse click and the preceding one,
198             // and then using that as the translation measure for the node in question. That scheme
199             // failed, however, and so did my next
200             // attempt which was to convert the mouse positions into the local coordinate system for the
201             // clicked node and proceed as before.
202             // This failed irrespective of whether I used the globalToLocal methods of the node or the
203             // cameraToLocal methods of the camera, which
204             // is hardly surprising given that the latter uses calls to the former to fulfill its purpose
205             . Finally, I tried using the inverse transform
206             // of the camera directly, i.e. without intermediary function calls and that did the trick !
207             Point globalPoint = new Point();
208             getCamera().getInverseViewTransform().transform( zme.getPoint(), globalPoint );
209             float x = (float)globalPoint.getX();
210             float y = (float)globalPoint.getY();

```

```

202         currentlyClickedNode.editor().getTransformGroup().setTranslation( x, y );
203     }
204   }
205 }
206 }

208 /** Handler for the pop-up menu activated by right clicking documents in the panorama
209 */
210 public class PopUpActionListener implements ActionListener
211 {
212   public void actionPerformed( ActionEvent ae )
213   {
214     JMenuItem menuItem = null;
215     // the below test makes explicit what is implicitly assumed: that the originator of the event
216     // was a popup menu item
217     if ( ae.getSource() instanceof JMenuItem )
218     {
219       menuItem = (JMenuItem) ae.getSource();
220     }
221     else
222     {
223       return;
224     }
225     getDocumentMenu().setVisible( false );
226     if ( menuItem == getResizeItem() )
227     {
228       // System.out.println( "resize" );
229       resizeDocument();
230     }
231     else if ( menuItem == getMoveItem() )
232     {
233       // System.out.println( "move" );
234       moveDocument();
235     }
236     /* if ( menuItem == getZoom() )
237     {
238       // System.out.println( "zoom" );
239       zoomDocument();
240     }
241     */ else if ( menuItem == getCopyTextItem() )
242     {
243       // System.out.println( "copy text" );
244       owner.getClientManager().copyText( currentlyClickedNode );
245     }
246     else if ( menuItem == getIvjLockItem() )
247     {
248       // System.out.println( "lock item" );
249       lockNode( currentlyClickedNode );
250     }
251     else if ( menuItem == getIvjUnlockAllItem() )
252     {
253       // System.out.println(" unlockall item" );
254       unLockAll();
255     }
256     else if ( menuItem == getIvjAutoLockAllItem() )
257     {
258       // System.out.println(" autolockall item" );
259       autoLock();
260     }
261     else if ( menuItem == getIvjArrangeAllItem() )
262     {
263       // System.out.println(" arrangeall item" );
264       validateLayout();
265       zoomToOverview();
266     }
267   }

```

```

    }

268     public class PopUpMenuMouseHandler extends MouseAdapter
269     {
270         public void mouseExited( MouseEvent me )
271         {
272             // The below was commented out in favour of a MouseMotionListener.mouseMoved event handler (
273             // below)
274             // The reason was that mouseExited events didn't seem to be generated when the mouse cursor was
275             // moved quickly off
276             // the source component
277             // getDocumentMenu().setVisible( false );
278         }
279     }

280     PanoramaMouseListener panoramaMouseListener = new PanoramaMouseListener();
281     PanoramaMouseMotionListener panoramaMouseMotionListener = new PanoramaMouseMotionListener();
282     float initialScale = 0;
283     ZVisualLeaf newLeaf;
284     ZPolyline zoomVector;
285     static Point previousMousePosition = null;
286     private ClientApplication owner;
287     private java.util.Hashtable documents = new Hashtable();
288     private java.util.Hashtable localDocuments = new Hashtable();
289     // value modified in popUpPopUpMenu and popDownPopUpMenu
290     private edu.umd.cs.jazz.ZVisualLeaf currentlyClickedNode = null;
291     private boolean movingNode = false;
292     private boolean zoomingNode = false;
293     private edu.umd.cs.jazz.util.ZBounds boundsBeforeResizing = null;
294     private boolean resizingNode = false;
295     private JSeparator ivjJSeparator2 = null;
296     private JMenuItem ivjMoveItem = null;
297     private JMenuItem ivjResizeItem = null;
298     private JPopupMenu ivjDocumentMenu = null;
299     private JMenuItem ivjCopyTextItem = null;
300     private javax.swing.Timer updateTimer = null;
301     private ClientPanoramaLayoutManager layoutManager = null;
302     private boolean panoramaLock = false;
303     private javax.swing.JMenuItem ivjUnlockAllItem = null;
304     private javax.swing.JCheckBoxMenuItem ivjAutoLockAllItem = null;
305     private javax.swing.JSeparator ivjSeparator3 = null;
306     private javax.swing.JSeparator ivjSeparator4 = null;
307     private javax.swing.JMenuItem ivjArrangeAllItem = null;
308     private javax.swing.JCheckBoxMenuItem ivjLockItem = null;

    /**
310     * ClientPanorama constructor comment.
311     */
312     public ClientPanorama()
313     {
314         super();
315         initialize();
316     }
317     /**
318      * ClientPanorama constructor comment.
319      * @param aRoot edu.umd.cs.jazz.ZRoot
320      * @param layer edu.umd.cs.jazz.ZLayerGroup
321      */
322     public ClientPanorama(edu.umd.cs.jazz.ZRoot aRoot, edu.umd.cs.jazz.ZLayerGroup layer)
323     {
324         super(aRoot, layer);
325     }
326     /**
327      * Add a new document to the panorama by inserting a leaf node and decorating it with the properties of
328      * the document, including its contents (passed as a byte arry).
329      * Creation date: (13-08-00 01:01:23)
330      * @return edu.umd.cs.jazz.ZVisualLeaf
331      * @param rawFileContents byte[]
332      * @param senderName java.lang.String

```

```

330 * @param authorName java.lang.String
331 * @param documentName java.lang.String
332 */
333 private ZVisualLeaf addDocument(byte[] rawFileContents, String senderName, String authorName, String
334 documentNameAndPath, String documentName)
335 {
336     ZVisualLeaf leaf;
337     ZTransformGroup transform;
338     PZSwing swing;
339     JScrollPane pane;
340
341     //System.out.println( "Document name " + documentName + "\n");
342     //System.out.println( "Document name and path " + documentNameAndPath + "\n");
343     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
344     String extension = fqPath.substring( fqPath.lastIndexOf('.') ).toLowerCase();
345     String mediaType = (String)ClientConstants.getMEDIA_TYPES().get(extension);
346     if ( mediaType == null )
347     {
348         mediaType = ClientConstants.getAll_FILES();
349     }
350     leaf = new ZVisualLeaf();
351     leaf.putClientProperty( ClientConstants.getAUTHOR_NAME(), authorName );
352     leaf.putClientProperty( ClientConstants.getDocument_NAME(), documentNameAndPath );
353     leaf.putClientProperty( ClientConstants.getSENDER_NAME(), senderName );
354     if (mediaType.equalsIgnoreCase( ClientConstants.getPLAIN_TEXT() ))
355     {
356         JTextArea textArea = new JTextArea( new String( rawFileContents ) );
357         textArea.setEditable( false );
358         pane = new JScrollPane( textArea );
359     }
360     else if (mediaType.equalsIgnoreCase( ClientConstants.getHTML() ))
361     {
362         JEditorPane editorPane = new JEditorPane( ClientConstants.getHTML(), new String( rawFileContents )
363             );
364         editorPane.setEditable( false );
365         pane = new JScrollPane( editorPane );
366     }
367     else if (mediaType.equalsIgnoreCase( ClientConstants.getRTF() ))
368     {
369         JEditorPane editorPane = new JEditorPane( ClientConstants.getRTF(), new String( rawFileContents ) );
370         editorPane.setEditable( false );
371         pane = new JScrollPane( editorPane );
372     }
373     else if (mediaType.equalsIgnoreCase( ClientConstants.getBITMAP() ))
374     {
375         // the unformatted byte array containing the image data is stored separately since extracting it
376         // from the
377         // ImageIcon after construction appears to be impossible - may be changed if I figure out a way
378         // that doesn't
379         // incur a hefty performance penalty
380         pane = new JScrollPane( new JLabel( new ImageIcon( rawFileContents ) ) );
381         pane.putClientProperty( ClientConstants.getRAW_BITMAP(), rawFileContents );
382     }
383     // default case
384     else
385     {
386         if ( rawFileContents.length < ClientConstants.getText_DOCUMENT_THRESHOLD() )
387         {
388             JTextArea textArea = new JTextArea( new String( rawFileContents ) );
389             textArea.setEditable( false );
390             pane = new JScrollPane( textArea );
391         }
392         else
393         {
394             pane = new JScrollPane( new JLabel( new ImageIcon( rawFileContents ) ) );
395         }
396     }
397     // add informative label to top of scroll pane, complete with bevelled border

```

```

392     JTextArea newLabel = newDocumentLabel( ClientConstants.getDOCUMENT_LABEL( documentNameAndPath,
393         authorName, rawFileContents.length, new java.util.Date() ) );
394     pane.setColumnHeaderView( newLabel );
395     pane.setPreferredSize( getLayoutManager().getDocumentSize() );
396     pane.setMaximumSize( pane.getPreferredSize() );
397     swing = new PZSwing(this, pane);
398     pane.scrollRectToVisible( pane.getViewportBorderBounds() );
399     leaf.setVisualComponent( swing );
400     leaf.putClientProperty( ClientConstants.getDOCUMENT_PATH(), fqPath );
401     // KAN OPTIMERES: det er ikke øndvendigt med en instans til hver node - øprv evt. med en enkelt, delt
402     // instans til alle nodes
403     // catch mouse events pertaining to this node and process them appropriately by adding a custom mouse
404     // listener
405     leaf.addMouseListener( panoramaMouseListener );
406     leaf.addMouseMotionListener( panoramaMouseMotionListener );
407     owner.addDocumentToVisibleList( senderName, authorName, documentNameAndPath, documentName );
408     getLayer().addChild(leaf);
409     return leaf;
410 }
411 /**
412  * Add document originating from other group members than the one using the client making the call to
413  * this function.
414  * Creation date: (13-08-00 00:58:00)
415  * @param rawFileContents byte[]
416  * @param senderName java.lang.String
417  * @param authorName java.lang.String
418  * @param documentName java.lang.String
419  */
420 public ZVisualLeaf addGlobalDocument( byte[] rawFileContents, String senderName, String authorName,
421     String documentNameAndPath, String documentName )
422 {
423     ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath,
424         documentName );
425     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
426     getDocuments().put( fqPath, newLeaf );
427     return newLeaf;
428 }
429 /**
430  * Add document added by the user of the client calling this method.
431  * Creation date: (13-08-00 01:05:09)
432  * @param rawFileContents byte[]
433  * @param senderName java.lang.String
434  * @param authorName java.lang.String
435  * @param documentName java.lang.String
436  */
437 public ZVisualLeaf addLocalDocument( byte[] rawFileContents, String senderName, String authorName, String
438     documentNameAndPath, long lastModified )
439 {
440     String documentName = documentNameAndPath.substring( documentNameAndPath.lastIndexOf( File.separator
441         ) + 1 );
442     ZVisualLeaf newLeaf = addDocument( rawFileContents, senderName, authorName, documentNameAndPath,
443         documentName );
444     newLeaf.putClientProperty( ClientConstants.getLAST_MODIFIED(), new Long( lastModified ) );
445     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentNameAndPath );
446     getLocalDocuments().put( fqPath, newLeaf );
447     return newLeaf;
448 }
449 /**
450  * Insert the method's description here.
451  * Creation date: (14-10-00 10:39:16)
452  */
453 public void autoLock()
454 {
455     if ( getIvjAutoLockAllItem().getState() == true )
456     {

```

```

448     ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.
        getTRUE() );
450   }
451   else
452   {
453     ClientInfo.getPreferences().setProperty( ClientConstants.getAUTOLOCK_NODES(), ClientConstants.
        getFALSE() );
454   }
455 /**
456 * Begin a zoom and pan sequence to have the node passed as argument centered and request the topic tree
457 * corresponding
458 * to that node from the server.
459 * Creation date: (07-08-00 09:19:29)
460 * @param zme edu.umd.cs.jazz.event.ZMouseEvent
461 */
462 public void centerDocument( ZVisualLeaf clickedNode )
463 {
464   String path = (String) clickedNode.getClientProperty( ClientConstants.getDocument_PATH() );
465   if ( ( path.equals( owner.getClientManager().getClientInfo().getCurrentDocumentName() ) == false ) )
466   {
467     owner.getClientManager().requestTopicTree( path );
468     //System.out.println( "Clicked path: " + path );
469   }
470   zoomAndPanToNode( clickedNode );
471 }
472 /**
473 * Convenience method to allow call by documentID rather than node
474 * Creation date: (30-08-00 23:51:07)
475 * @param documentID java.lang.String
476 */
477 public void centerDocument(String documentID)
478 {
479   if ( getLocalDocuments().containsKey( documentID ) )
480   {
481     centerDocument( (ZVisualLeaf) getLocalDocuments().get( documentID ) );
482   }
483   else
484   {
485     centerDocument( (ZVisualLeaf) getDocuments().get( documentID ) );
486   }
487 }
488 /**
489 * 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
491 * translated, although correctly sized, bounding box. The below use of getBounds followed by a
492 * localToGlobal transform seems to produce
493 * a correct result. Maybe the effect is due to value caching or spurious translations by function
494 * getGlobalBounds()...
495 * Creation date: (12-10-00 23:59:36)
496 * @return java.awt.Rectangle
497 * @param nodes edu.umd.cs.jazz.ZNode[]
498 */
499 public java.awt.Rectangle computeOverviewRectangle(ZNode[] nodes)
500 {
501   /* float minX = Float.MAX_VALUE, minY = Float.MAX_VALUE, maxX = Float.MIN_VALUE, maxY = Float.MIN_VALUE;
502   float xcandidate, ycandidate;
503   for ( int i = 0 ; i < nodes.length; i++ )
504   {
505     ZBounds bounds = nodes[i].getBounds();
506     nodes[i].localToGlobal( bounds );
507     xcandidate = (int)bounds.getMinX();
508     ycandidate = (int)bounds.getMinY();
509     minX = Math.abs( xcandidate )< Math.abs( minX )? xcandidate : minX;
510     minY = Math.abs( ycandidate )< Math.abs( minY )? ycandidate : minY;

```

```

510     xcandidate = (int)bounds.getMaxX();
511     ycandidate = (int)bounds.getMaxY();
512     maxX = Math.abs( xcandidate ) > Math.abs( maxX ) ? xcandidate : maxX;
513     maxY = Math.abs( ycandidate ) > Math.abs( maxY ) ? ycandidate : maxY;
514 }
514 java.awt.Rectangle overviewRectangle = new java.awt.Rectangle( (int)minX, (int)minY, (int)maxX, (int)
515     maxY );
516     return overviewRectangle;
516 */
516     return (java.awt.Rectangle) getLayer().getGlobalBounds().getBounds();
518 }
518 /**
520 * Remove the panorama pop-up menu.
520 * Creation date: (20-07-00 17:59:56)
522 */
522 public void dismissPopupMenu()
524 {
524     getDocumentMenu().setVisible( false );
526 }
526 /**
528 * Perform a resize operation.
528 * Creation date: (31-07-00 20:02:40)
530 * @param zMouseEvent edu.umd.cs.jazz.event.ZMouseEvent
530 */
532 public void doResize(ZMouseEvent zme)
532 {
534     // if right click, return to original size
534     if ( zme.isMetaDown() == true )
536     { // use bounds of resize rectangle to reshape active scenegraph node and notify relevant objects
536         // that this has been done
537         Dimension newSize = new Dimension();
538         //getCamera().cameraToLocal( bounds, currentlyClickedNode );
539         newSize.setSize(
540             (int) boundsBeforeResizing.getMaxX() - (int) boundsBeforeResizing.getMinX(),
540             (int) boundsBeforeResizing.getMaxY() - (int) boundsBeforeResizing.getMinY() );
542         ( (PZSwing)currentlyClickedNode.getVisualComponent() ).getComponent().setPreferredSize( newSize );
542         ( (PZSwing)currentlyClickedNode.getVisualComponent() ).computeBounds();
544         currentlyClickedNode.reshape();
544     }
546     revalidate();
546     getDrawingSurface().repaint();
548     setResizingNode( false );
548 }
550 /**
550 * Carry out a zooming operation
552 * Creation date: (31-07-00 21:05:01)
552 */
554 public void doZooming( ZMouseEvent zme )
554 {
556     if ( zme.isMetaDown() == true )
556     {
558         currentlyClickedNode.editor().getTransformGroup().setScale( initialScale );
558     }
560     previousMousePosition = null;
560     setZoomingNode( false );
562     ( (PZSwing) currentlyClickedNode.getVisualComponent() ).setZoomingNode( null );
562     // getLayer().removeChild( newLeaf );
564     revalidate();
564     getDrawingSurface().repaint();
566 }
566 /**
568 * Insert all panorama documents, both local and global into an array and return it.
568 * Creation date: (14-10-00 16:42:03)
570 * @return edu.umd.cs.jazz.ZNode[]
570 */
572 public ZNode[] getAllDocumentsAsArray()

```

```

    {
574     ZNode nodes[] = new ZNode[ getLocalDocuments().size() + getDocuments().size() ];
575     int i = 0;
576     for (int j = 0 ; j < getLocalDocuments().size(); j++, i++ )
577     {
578         nodes[ i ] = (ZNode) getLocalDocuments().values().toArray()[ j ];
579     }
580     for ( int j = 0 ; j < getDocuments().size(); j++, i++ )
581     {
582         nodes[ i ] = (ZNode) getDocuments().values().toArray()[ j ];
583     }
584     return nodes;
585 }
586 /**
587 * Insert the method's description here.
588 * Creation date: (31-07-00 19:43:21)
589 * @return edu.umd.cs.jazz.util.ZBounds
590 */
591 public edu.umd.cs.jazz.util.ZBounds getBoundsBeforeResizing() {
592     return boundsBeforeResizing;
593 }
594 /**
595 * Return the CopyText property value.
596 * @return javax.swing.JMenuItem
597 */
598 /* WARNING: THIS METHOD WILL BE REGENERATED. */
599 private javax.swing.JMenuItem getCopyTextItem() {
600     if (ivjCopyTextItem == null) {
601         try {
602             ivjCopyTextItem = new javax.swing.JMenuItem();
603             ivjCopyTextItem.setName("CopyTextItem");
604             ivjCopyTextItem.setMnemonic('C');
605             ivjCopyTextItem.setText("Copy\u00d7text");
606             // user code begin {1}
607             ivjCopyTextItem.addActionListener( new PopUpActionListener() );
608             // user code end
609         } catch (java.lang.Throwable ivjExc) {
610             // user code begin {2}
611             // user code end
612             handleException(ivjExc);
613         }
614     }
615     return ivjCopyTextItem;
616 }
617 /**
618 * Insert the method's description here.
619 * Creation date: (28-07-00 00:50:07)
620 * @return edu.umd.cs.jazz.ZVisualLeaf
621 */
622 public edu.umd.cs.jazz.ZVisualLeaf getCurrentlyClickedNode() {
623     return currentlyClickedNode;
624 }
625 /**
626 * Helper method to return a document node, irrespective of whether it is global or local
627 * Creation date: (14-10-00 11:47:11)
628 * @return edu.umd.cs.jazz.ZVisualLeaf
629 * @param clientName java.lang.String
630 * @param documentName java.lang.String
631 */
632 public ZVisualLeaf getDocument(String clientName, String documentName)
633 {
634     String fqName = ClientConstants.createFullyQualifiedDocumentName( clientName, documentName );
635     ZVisualLeaf returnLeaf = null;
636     if ( localDocuments.get( fqName )!= null )
637     {
638         returnLeaf = (ZVisualLeaf) getLocalDocuments().get( fqName );

```

```

        }
640    else
641    {
642        returnLeaf = (ZVisualLeaf) getDocuments().get( fqName );
643    }
644    return returnLeaf;
645 }
646 /**
647 * Return the PopupMenu property value.
648 * @return javax.swing.JPopupMenu
649 */
650 /* WARNING: THIS METHOD WILL BE REGENERATED. */
651 private JPopupMenu getDocumentMenu() {
652     if (ivjDocumentMenu == null) {
653         try {
654             ivjDocumentMenu = new JPopupMenu();
655             ivjDocumentMenu.setName("DocumentMenu");
656             ivjDocumentMenu.add(getResizeItem());
657             ivjDocumentMenu.add(getMoveItem());
658             ivjDocumentMenu.add(getIvjLockItem());
659             ivjDocumentMenu.add(getJSeparator2());
660             ivjDocumentMenu.add(getCopyTextItem());
661             ivjDocumentMenu.add(getIvjSeparator3());
662             ivjDocumentMenu.add(getIvjUnlockAllItem());
663             ivjDocumentMenu.add(getIvjArrangeAllItem());
664             ivjDocumentMenu.add(getIvjAutoLockAllItem());
665             // user code begin {1}
666             // user code end
667         } catch (java.lang.Throwable ivjExc) {
668             // user code begin {2}
669             // user code end
670             handleException(ivjExc);
671         }
672     }
673     return ivjDocumentMenu;
674 }
675 /**
676 * Insert the method's description here.
677 * Creation date: (08-07-00 01:30:08)
678 * @return java.util.Hashtable
679 */
680 public java.util.Hashtable getDocuments() {
681     return documents;
682 }
683 /**
684 * Insert the method's description here.
685 * Creation date: (14-10-00 13:35:55)
686 * @return javax.swing.JMenuItem
687 */
688 public javax.swing.JMenuItem getIvjArrangeAllItem()
689 {
690     if (ivjArrangeAllItem == null) {
691         try {
692             ivjArrangeAllItem = new javax.swing.JMenuItem();
693             ivjArrangeAllItem.setName("ArrangeAllItem");
694             ivjArrangeAllItem.setMnemonic('R');
695             ivjArrangeAllItem.setText("Arrange\u2022all");
696             // user code begin {1}
697             ivjArrangeAllItem.addActionListener( new PopUpActionListener() );
698             // user code end
699         } catch (java.lang.Throwable ivjExc) {
700             // user code begin {2}
701             // user code end
702             handleException(ivjExc);
703         }
704     }

```

```

        return ivjArrangeAllItem;
706    }
708 /**
709  * Insert the method's description here.
710  * Creation date: (14-10-00 09:53:08)
711  * @return javax.swing.JCheckBoxMenuItem
712  */
713 public javax.swing.JCheckBoxMenuItem getIvjAutoLockAllItem() {
714     if (ivjAutoLockAllItem == null) {
715         try {
716             ivjAutoLockAllItem = new javax.swing.JCheckBoxMenuItem();
717             ivjAutoLockAllItem.setName("AutoLockAllItem");
718             ivjAutoLockAllItem.setMnemonic('A');
719             ivjAutoLockAllItem.setText("Auto-lock>All");
720             // user code begin {1}
721             ivjAutoLockAllItem.addActionListener( new PopUpActionListener() );
722             // user code end
723         } catch (java.lang.Throwable ivjExc) {
724             // user code begin {2}
725             // user code end
726             handleException(ivjExc);
727         }
728     }
729     return ivjAutoLockAllItem;
730 }
731 /**
732  * Insert the method's description here.
733  * Creation date: (14-10-00 13:42:49)
734  * @return javax.swing.JCheckBoxMenuItem
735  */
736 public javax.swing.JCheckBoxMenuItem getIvjLockItem()
737 {
738     if (ivjLockItem == null) {
739         try {
740             ivjLockItem = new javax.swing.JCheckBoxMenuItem();
741             ivjLockItem.setName("LockItem");
742             ivjLockItem.setMnemonic('L');
743             ivjLockItem.setText("Lock");
744             // user code begin {1}
745             ivjLockItem.addActionListener( new PopUpActionListener() );
746             // user code end
747         } catch (java.lang.Throwable ivjExc) {
748             // user code begin {2}
749             // user code end
750             handleException(ivjExc);
751         }
752     }
753     return ivjLockItem;
754 }
755 /**
756  * Insert the method's description here.
757  * Creation date: (14-10-00 10:14:35)
758  * @return javax.swing.JSeparator
759  */
760 public javax.swing.JSeparator getIvjSeparator3() {
761     if (ivjSeparator3 == null) {
762         try {
763             ivjSeparator3 = new javax.swing.JSeparator();
764             ivjSeparator3.setName("JSeparator3");
765             // user code begin {1}
766             // user code end
767         } catch (java.lang.Throwable ivjExc) {
768             // user code begin {2}
769             // user code end
770         }
771     }
772     return ivjSeparator3;
773 }

```

```

        handleException(ivjExc);
772     }
773 }
774 return ivjSeparator3;
775 }
776 /**
 * Insert the method's description here.
777 * Creation date: (14-10-00 10:14:58)
778 * @return javax.swing.JSeparator
779 */
780 public javax.swing.JSeparator getIvjSeparator4() {
781     if (ivjSeparator4 == null) {
782         try {
783             ivjSeparator4 = new javax.swing.JSeparator();
784             ivjSeparator4.setName("JSeparator4");
785             // user code begin {1}
786             // user code end
787         } catch (java.lang.Throwable ivjExc) {
788             // user code begin {2}
789             // user code end
790             handleException(ivjExc);
791         }
792     }
793     return ivjSeparator4;
794 }
795 }
796 /**
797 * Insert the method's description here.
798 * Creation date: (14-10-00 09:51:30)
799 * @return javax.swing.JMenuItem
800 */
801 public javax.swing.JMenuItem getIvjUnlockAllItem() {
802     if (ivjUnlockAllItem == null) {
803         try {
804             ivjUnlockAllItem = new javax.swing.JMenuItem();
805             ivjUnlockAllItem.setName("UnlockAllItem");
806             ivjUnlockAllItem.setMnemonic('N');
807             ivjUnlockAllItem.setText("Unlock>All");
808             // user code begin {1}
809             ivjUnlockAllItem.addActionListener( new PopUpActionListener() );
810             // user code end
811         } catch (java.lang.Throwable ivjExc) {
812             // user code begin {2}
813             // user code end
814             handleException(ivjExc);
815         }
816     }
817     return ivjUnlockAllItem;
818 }
819 }
820 /**
821 * Return the JSeparator2 property value.
822 * @return javax.swing.JSeparator
823 */
824 /**
825 * WARNING: THIS METHOD WILL BE REGENERATED. */
826 private javax.swing.JSeparator getJSeparator2() {
827     if (ivjJSeparator2 == null) {
828         try {
829             ivjJSeparator2 = new javax.swing.JSeparator();
830             ivjJSeparator2.setName("JSeparator2");
831             // user code begin {1}
832             // user code end
833         } catch (java.lang.Throwable ivjExc) {
834             // user code begin {2}
835             // user code end
836             handleException(ivjExc);

```

```

        }
838     }
839     return ivjJSeparator2;
840 }
841 /**
842  * Insert the method's description here.
843  * Creation date: (29-08-00 01:03:17)
844  * @return clientapp.ClientPanoramaLayout
845  */
846 public ClientPanoramaLayoutManager getLayoutManager() {
847     if( layoutManager == null )
848     {
849         setLayoutManager( new GridPanoramaLayoutManager() );
850     }
851     return layoutManager;
852 }
853 /**
854  * Insert the method's description here.
855  * Creation date: (26-07-00 11:38:26)
856  * @return java.util.Hashtable
857  */
858 public java.util.Hashtable getLocalDocuments()
859 {
860     return localDocuments;
861 }
862 /**
863  * Return the MoveItem property value.
864  * @return javax.swing.JMenuItem
865  */
866 /* WARNING: THIS METHOD WILL BE REGENERATED. */
867 private javax.swing.JMenuItem getMoveItem() {
868     if (ivjMoveItem == null) {
869         try {
870             ivjMoveItem = new javax.swing.JMenuItem();
871             ivjMoveItem.setName("MoveItem");
872             ivjMoveItem.setMnemonic('M');
873             ivjMoveItem.setText("Move");
874             // user code begin {1}
875             ivjMoveItem.addActionListener( new PopUpActionListener() );
876             // user code end
877         } catch (java.lang.Throwable ivjExc) {
878             // user code begin {2}
879             // user code end
880             handleException(ivjExc);
881         }
882     }
883     return ivjMoveItem;
884 }
885 /**
886  * Insert the method's description here.
887  * Creation date: (18-06-00 17:41:38)
888  * @return clientapp.ClientApplication
889  */
890 public ClientApplication getOwner() {
891     return owner;
892 }
893 /**
894  * Return the ResizeItem property value.
895  * @return javax.swing.JMenuItem
896  */
897 /* WARNING: THIS METHOD WILL BE REGENERATED. */
898 private javax.swing.JMenuItem getResizeItem() {
899     if (ivjResizeItem == null) {
900         try {
901             ivjResizeItem = new javax.swing.JMenuItem();
902             ivjResizeItem.setName("ResizeItem");

```

```

        ivjResizeItem.setMnemonic('R');
904    ivjResizeItem.setText("Resize");
905    // user code begin {1}
906    ivjResizeItem.addActionListener( new PopUpActionListener() );
907    // user code end
908    } catch (java.lang.Throwable ivjExc) {
909        // user code begin {2}
910        // user code end
911        handleException(ivjExc);
912    }
913 }
914 return ivjResizeItem;
}
915 /**
 * Insert the method's description here.
916 * Creation date: (22-08-00 15:44:15)
917 * @return javax.swing.Timer
918 */
919 public javax.swing.Timer getUpdateTimer()
920 {
921     if ( updateTimer == null)
922     { // start update process in separate thread - multiply by 1000 to get the preferred or default
923         setting in milliseconds.
924         updateTimer = new Timer( Integer.parseInt( ClientInfo.getPreferences().getProperty(
925             ClientConstants.UPDATE_INTERVAL() ))* 1000,
926             new UpdateActionListener() );
927     }
928     return updateTimer;
}
929 /**
 * Called whenever the part throws an exception.
930 * @param exception java.lang.Throwable
 */
931 private void handleException(java.lang.Throwable exception) {
932
933     /* Uncomment the following lines to print uncaught exceptions to stdout */
934     // System.out.println("----- UNCAUGHT EXCEPTION -----");
935     // exception.printStackTrace(System.out);
}
936 /**
 * Insert the method's description here.
937 * Creation date: (16-06-00 17:44:24)
 */
938 public void initialize()
{
939     try {
940         // user code begin {1}
941         // user code end
942         setName("ClientPanorama");
943         setSize(1, 1);
944     } catch (java.lang.Throwable ivjExc) {
945         handleException(ivjExc);
}
946     // user code begin {2}
947     getCameraNode().addMouseListener( new CameraMouseListener() );
948     getCameraNode().addMouseMotionListener( new PanoramaMouseMotionListener() );
949     getCamera().addCameraListener( new CameraListener() );
950     zoomEventHandler = new PZoomHandler( getCameraNode() );
951     zoomEventHandler.setActive( true );
952     panEventHandler = new PZPanEventHandler( getCameraNode() );
953     panEventHandler.setActive( true );
954     swingEventHandler = new ZSwingEventHandler( this, getCameraNode() );
955     swingEventHandler.setActive( true );
956     getDrawingSurface().setRenderQuality( Integer.parseInt( (String) ClientConstants.getDISPLAY_QUALITIES
());

```

```

                get( ClientInfo.getPreferences().getProperty( ClientConstants.
                        getDISPLAY_QUALITY() )));

966     getUpdateTimer().start();
967     getLayoutManager().setSettings( ClientInfo.getPreferences() );
968     //getLayer().addGroupListener( new UpdateHandler() );
969     // user code end
970 }

971 /**
972  * Insert the method's description here.
973  * Creation date: (29-07-00 18:22:23)
974  * @return boolean
975  */
976 public boolean isMovingNode() {
977     return movingNode;
978 }
979 /**
980  * Insert the method's description here.
981  * Creation date: (07-09-00 21:16:49)
982  * @return boolean
983  */
984 public boolean isPanoramaLock() {
985     return panoramaLock;
986 }
987 /**
988  * Insert the method's description here.
989  * Creation date: (31-07-00 20:04:03)
990  * @return boolean
991  */
992 public boolean isResizingNode() {
993     return resizingNode;
994 }
995 /**
996  * Insert the method's description here.
997  * Creation date: (29-07-00 19:19:43)
998  * @return boolean
999  */
1000 public boolean isZoomingNode() {
1001     return zoomingNode;
1002 }
1003 /**
1004  * Insert the method's description here.
1005  * Creation date: (14-10-00 10:37:47)
1006  * @param node edu.umd.cs.jazz.ZVisualLeaf
1007  */
1008 public void lock(ZNode node)
1009 {
1010     node.putClientProperty( ClientConstants.getNODE_LOCKED(), ClientConstants.getTRUE() );
1011 }
1012 /**
1013  * Insert the method's description here.
1014  * Creation date: (14-10-00 13:47:46)
1015  * @param node edu.umd.cs.jazz.ZNode
1016  */
1017 public void lockNode(ZNode node)
1018 {
1019     if ( getIvjLockItem().getState() == true )
1020     {
1021         lock( node );
1022     }
1023     else
1024     {
1025         unlock( node );
1026     }
1027 }
1028 /**
1029  * main entrypoint - starts the part when it is run as an application

```

```

1030 * @param args java.lang.String[]
1031 */
1032 public static void main(java.lang.String[] args) {
1033     try {
1034         JFrame frame = new javax.swing.JFrame();
1035         ClientPanorama aClientPanorama;
1036         aClientPanorama = new ClientPanorama();
1037         frame.setContentPane(aClientPanorama);
1038         frame.setSize(aClientPanorama.getSize());
1039         frame.addWindowListener(new java.awt.event.WindowAdapter() {
1040             public void windowClosing(java.awt.event.WindowEvent e) {
1041                 System.exit(0);
1042             }
1043         });
1044         frame.setVisible(true);
1045     } catch (Throwable exception) {
1046         System.err.println("Exception occurred in main() of clientapp.ClientPanorama");
1047         exception.printStackTrace(System.out);
1048     }
1049 }
1050 /**
1051 * Translate the document associated with the currently active node in the direction and by the amount
1052 * indicated by the
1053 * user's mouse movements.
1054 * Creation date: (28-07-00 01:13:44)
1055 */
1056 public void moveDocument()
1057 {
1058     setMovingNode( true );
1059     previousMousePosition = null;
1060 }
1061 /**
1062 * Insert the method's description here.
1063 * Creation date: (22-08-00 11:18:54)
1064 * @return javax.swing.JTextArea
1065 * @param text java.lang.String
1066 */
1067 public JTextArea newDocumentLabel(String text)
1068 {
1069     JTextArea newLabel = new JTextArea( text );
1070     newLabel.setEditable( false );
1071     newLabel.setBackground( Color.orange );
1072     // newLabel.setSize( new Dimension( Integer.parseInt( ClientConstants.getDEFAULT_DOCUMENT_WIDTH() ), (
1073     int ) newLabel.getSize().getHeight() ) );
1074     Border bevelBorder = BorderFactory.createRaisedBevelBorder();
1075     Border matteBorder = BorderFactory.createMatteBorder( ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1076                                         ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1077                                         ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1078                                         ClientConstants.getDOCUMENT_BORDER_WIDTH(),
1079                                         newLabel.getBackground() );
1080     Border compoundBorder = BorderFactory.createCompoundBorder( bevelBorder, matteBorder );
1081     newLabel.setBorder( compoundBorder );
1082     newLabel.setLineWrap( true );
1083     return newLabel;
1084 }
1085 /**
1086 * Insert the method's description here.
1087 * Creation date: (28-07-00 01:02:00)
1088 */
1089 public void popDownPopupMenu()
1090 {
1091     getDocumentMenu().setVisible( false );
1092 // setCurrentlyClickedNode( null );
1093 }
1094 /**

```

```

    * 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 );
1099     JScrollPane clickedPane = (JScrollPane) ( (PZSwing) node.getVisualComponent() ).getComponent();
1100     // enable text copying only if the clicked node displays text, either formatted or plain
1101     if ( clickedPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1102     {
1103         getCopyTextItem().setEnabled( true );
1104     }
1105     else
1106     {
1107         getCopyTextItem().setEnabled( false );
1108     }
1109     boolean state = ClientInfo.getPreferences().getProperty( ClientConstants.getAUTOLOCK_NODES() ).equals
1110         ( ClientConstants.getTRUE() )? true : false;
1111     getIvjAutoLockAllItem().setState( state );
1112     if ( getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() )!= null )
1113     {
1114         state = getCurrentlyClickedNode().getClientProperty( ClientConstants.getNODE_LOCKED() ).equals(
1115             ClientConstants.getTRUE() )? true : false;
1116     }
1117     else
1118     {
1119         state = false;
1120     }
1121     getIvjLockItem().setState( state );
1122     getDocumentMenu().show( this, (int)position.getX(), (int)position.getY() );
1123     getDocumentMenu().setVisible( true );
1124 }
1125 /**
1126 * Remove document originating from other group member.
1127 * Creation date: (03-08-00 13:13:00)
1128 * @param path java.lang.String
1129 */
1130 public void removeGlobalDocument(String senderName, String documentName )
{
1131     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
1132     ZNode topNode = ( ZNode ) getDocuments().get( fqPath ).editor().getTop();
1133     if ( topNode != null )
1134     {
1135         topNode.getParent().removeChild( topNode );
1136     }
1137     owner.removeDocumentFromVisibleList( senderName, documentName );
1138     getDocuments().remove( fqPath );
1139 }
1140 /**
1141 * Remove document inserted by the user the client making the call to this method.
1142 * Creation date: (23-06-00 08:39:46)
1143 * @param path java.lang.String
1144 */
1145 public void removeLocalDocument(String senderName, String documentName )
{
1146     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
1147     ZNode topNode = ( ZNode ) getLocalDocuments().get( fqPath ).editor().getTop();
1148     if ( topNode != null )
1149     {
1150         topNode.getParent().removeChild( topNode );
1151         getLocalDocuments().remove( fqPath );
1152         // System.out.println( "Removed: " + fqPath );
1153     }
1154     owner.removeDocumentFromVisibleList( senderName, documentName );
1155 }

```

```

1156 /**
 * 
1158 * Creation date: (28-07-00 01:13:33)
 */
1160 public void resizeDocument()
{
1162     boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
     setResizingNode( true );
1164 }
1165 /**
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;
1223 }
1224 /**
1225 * Insert the method's description here.
1226 * Creation date: (14-10-00 13:42:49)
1227 * @param newIvjLockItem javax.swing.JCheckBoxMenuItem
1228 */
1229 void setIvjLockItem(javax.swing.JCheckBoxMenuItem newIvjLockItem) {
1230     ivjLockItem = newIvjLockItem;
1231 }
1232 /**
1233 * Insert the method's description here.
1234 * Creation date: (14-10-00 10:14:35)
1235 * @param newIvjSeparator3 javax.swing.JSeparator
1236 */
1237 void setIvjSeparator3(javax.swing.JSeparator newIvjSeparator3) {
1238     ivjSeparator3 = newIvjSeparator3;
1239 }
1240 /**
1241 * Insert the method's description here.
1242 * Creation date: (14-10-00 10:14:58)
1243 * @param newIvjSeparator4 javax.swing.JSeparator
1244 */
1245 void setIvjSeparator4(javax.swing.JSeparator newIvjSeparator4) {
1246     ivjSeparator4 = newIvjSeparator4;
1247 }
1248 /**
1249 * Insert the method's description here.
1250 * Creation date: (14-10-00 09:51:30)
1251 * @param newIvjUnlockAllItem javax.swing.JMenuItem
1252 */
1253 void setIvjUnlockAllItem(javax.swing.JMenuItem newIvjUnlockAllItem) {
1254     ivjUnlockAllItem = newIvjUnlockAllItem;
1255 }
1256 /**
1257 * Insert the method's description here.
1258 * Creation date: (29-08-00 01:03:17)
1259 * @param newLayoutManager clientapp.ClientPanoramaLayout
1260 */
1261 public void setLayoutManager(ClientPanoramaLayoutManager newLayoutManager) {
1262     layoutManager = newLayoutManager;
1263 }
1264 /**
1265 * Insert the method's description here.
1266 * Creation date: (26-07-00 11:38:26)
1267 * @param newLocalDocuments java.util.Hashtable
1268 */
1269 public void setLocalDocuments(java.util.Hashtable newLocalDocuments) {
1270     localDocuments = newLocalDocuments;
1271 }
1272 /**
1273 * Insert the method's description here.
1274 * Creation date: (29-07-00 18:22:23)
1275 * @param newMovingNode boolean
1276 */
1277 public void setMovingNode(boolean newMovingNode) {
1278     movingNode = newMovingNode;
1279 }
1280 /**
1281 * Insert the method's description here.
1282 * Creation date: (18-06-00 17:41:38)
1283 * @param newOwner clientapp.ClientApplication
1284 */
1285 public void setOwner(ClientApplication newOwner) {
1286     owner = newOwner;
1287 }

```

```

1288 /**
1289  * Insert the method's description here.
1290  * Creation date: (07-09-00 21:16:49)
1291  * @param newPanoramaLock boolean
1292 */
1293 public void setPanoramaLock(boolean newPanoramaLock) {
1294     panoramaLock = newPanoramaLock;
1295     PZSwing.setUpdating( newPanoramaLock );
1296 }
1297 /**
1298  * Insert the method's description here.
1299  * Creation date: (31-07-00 20:04:03)
1300  * @param newResizingNode boolean
1301 */
1302 public void setResizingNode(boolean newResizingNode) {
1303     resizingNode = newResizingNode;
1304 }
1305 /**
1306  * Insert the method's description here.
1307  * Creation date: (22-08-00 15:44:15)
1308  * @param newUpdateTimer javax.swing.Timer
1309 */
1310 public void setUpdateTimer(javax.swing.Timer newUpdateTimer) {
1311     updateTimer = newUpdateTimer;
1312 }
1313 /**
1314  * Insert the method's description here.
1315  * Creation date: (14-07-00 15:55:44)
1316  * @param zoomNode edu.umd.cs.jazz.ZNode
1317 */
1318 public void setZoomHandler(ZNode zoomNode)
1319 {
1320     zoomEventHandler = new PZoomHandler( zoomNode );
1321 }
1322 /**
1323  * Insert the method's description here.
1324  * Creation date: (29-07-00 19:19:43)
1325  * @param newZoomingNode boolean
1326 */
1327 public void setZoomingNode(boolean newZoomingNode) {
1328     zoomingNode = newZoomingNode;
1329 }
1330 /**
1331  * Insert the method's description here.
1332  * Creation date: (14-10-00 10:38:24)
1333  * @param node edu.umd.cs.jazz.ZVisualLeaf
1334 */
1335 public void unlock(ZNode node)
1336 {
1337     node.putClientProperty( ClientConstants.getNODE_LOCKED(), ClientConstants.getFALSE() );
1338 }
1339 /**
1340  * Insert the method's description here.
1341  * Creation date: (14-10-00 10:38:45)
1342 */
1343 public void unLockAll()
1344 {
1345     Iterator iterator = getDocuments().values().iterator();
1346     while ( iterator.hasNext() )
1347     {
1348         unlock( (ZVisualLeaf) iterator.next() );
1349     }
1350     iterator = getLocalDocuments().values().iterator();
1351     while ( iterator.hasNext() )
1352     {
1353         unlock( (ZVisualLeaf) iterator.next() );

```

```

1354     }
1355 }
1356 /**
 * Apply a difference array received from another group member to a document in the panorama.
1357 * Creation date: (21-08-00 17:42:03)
1358 * @param fqPath java.lang.String
1359 * @param newRawFileContents byte[]
1360 */
1361 public synchronized void updateDistributedDocument(String fqPath, byte[] differenceBitmap )
{
1362     byte[] oldRawFileContents = null;
1363     byte[] newRawFileContents = null;
1364     ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
1365     PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
1366     JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
1367     java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
1368     if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
1369     {
1370         oldRawFileContents = (byte[]) affectedNode.getClientProperty( ClientConstants.getRAW_BITMAP() );
1371         newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
1372             differenceBitmap );
1373         affectedScrollPane.setViewport( new JLabel( new ImageIcon( newRawFileContents ) ) );
1374         affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
1375     }
1376     else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1377     {
1378         oldRawFileContents = ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView()
1379             ).getText().getBytes();
1380         newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
1381             differenceBitmap );
1382         ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
1383             String( newRawFileContents ) );
1384     }
1385     affectedScrollPane.setColumnHeaderView( newDocumentLabel(
1386         ClientConstants.getDocument_LABEL( (String) affectedNode.getClientProperty( ClientConstants.
1387             getDOCUMENT_NAME() ),
1388             (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
1389                 ),
1390             newRawFileContents.length,
1391             new java.util.Date() ) );
1392     // ensure that the visible rectangle is unchanged
1393     affectedScrollPane.scrollRectToVisible( visibleRectangle );
1394     revalidate();
1395     getDrawingSurface().repaint();
1396 }
1397 /**
 * Variation on updateDistributedDocument(String, byte[])
1398 * Creation date: (21-08-00 17:42:03)
1399 * @param fqPath java.lang.String
1400 * @param newRawFileContents byte[]
1401 */
1402 public synchronized void updateDistributedDocument(String senderName, String documentName, byte[]
1403     differenceBitmap )
{
1404     byte[] oldRawFileContents = null;
1405     byte[] newRawFileContents = null;
1406     String fqPath = ClientConstants.createFullyQualifiedDocumentName( senderName, documentName );
1407     ZVisualLeaf affectedNode = (ZVisualLeaf) getDocuments().get( fqPath );
1408     // the update was of a document that is not present in the panorama. It should therefore be requested
1409     // again from
1410     // the update. This is now likely to succeed as the successful arrival of the update indicates that
1411     // the connection
1412     // is "live"
1413     if ( affectedNode == null )
1414     {

```

```

    return; // TO DO
1412 }

1414 PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
1415 JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
1416 java.awt.Rectangle visibleRectangle = affectedScrollPane.getViewport().getViewRect();
1417 if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
1418 {
1419     oldRawFileContents = (byte[]) affectedNode.getClientProperty( ClientConstants.getRAW_BITMAP() );
1420     newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
1421         differenceBitmap );
1422     affectedScrollPane.setViewport( new JLabel( new ImageIcon( newRawFileContents ) ) );
1423     affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
1424 }
1425 else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1426 {
1427     oldRawFileContents = ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView()
1428         ).getText().getBytes();
1429     newRawFileContents = owner.getClientManager().computeBitmapCombination( oldRawFileContents,
1430         differenceBitmap );
1431     ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
1432         String( newRawFileContents ) );
1433 }
1434 affectedScrollPane.setColumnHeaderView( newDocumentLabel(
1435     ClientConstants.getDocument_LABEL( (String) affectedNode.getClientProperty( ClientConstants.
1436         getDOCUMENT_NAME() ),
1437             (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
1438                 ),
1439             newRawFileContents.length,
1440             new java.util.Date() ) );
1441 // ensure that the visible rectangle is unchanged
1442 affectedScrollPane.scrollRectToVisible( visibleRectangle );
1443 revalidate();
1444 getDrawingSurface().repaint();

1445 }
1446 /**
1447 * Update the local documents displayed in the panorama as necessary.
1448 * Creation date: (21-08-00 12:11:47)
1449 */
1450 public synchronized void updateDocuments()
1451 {
1452     // safe because of short-circuit evaluation (latter part is evaluated only if the former is true)
1453     if ( ( getLocalDocuments() != null ) && ( getLocalDocuments().size() > 0 ) )
1454     {
1455         Collection values = getLocalDocuments().values();
1456         Iterator iterator = values.iterator();
1457         // only notify the user that updating is taking place if the frequency is not too great to avoid
1458         // visual clutter
1459         if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ) )
1460         {
1461             owner.initMessageArea( 1, values.size(), 1, ClientConstants.getUPDATING_DOCUMENTS() );
1462         }
1463         while ( iterator.hasNext() )
1464         {
1465             ZVisualLeaf documentNode = (ZVisualLeaf) iterator.next();
1466             File file = new File( (String) documentNode.getClientProperty( ClientConstants.getDocument_NAME()
1467                 ) );
1468             long lastModified = ( (Long) documentNode.getClientProperty( ClientConstants.getLast_MODIFIED()
1469                 ) ).longValue();
1470             if ( file.lastModified() != lastModified )
1471             {
1472                 documentNode.putClientProperty( ClientConstants.getLast_MODIFIED(), new Long( file.
1473                     lastModified() ) );
1474                 JScrollPane documentPane = (JScrollPane) ( (PZSwing) documentNode.getVisualComponent() ).
1475                     getComponent();

```

```

1466     byte rawFileContents[] = null;
1468
1469     if ( documentPane.getViewport().getView() instanceof JLabel )
1470     {
1471         rawFileContents = (byte[]) documentPane.getClientProperty( ClientConstants.getRAW_BITMAP()
1472                                         );
1473     }
1474     else if ( documentPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1475     {
1476         rawFileContents = ( (javax.swing.text.JTextComponent )documentPane.getViewport().getView()
1477                                         ).getText().getBytes();
1478     }
1479     try
1480     {
1481         byte newRawFileContents[] = owner.getClientManager().readDocument( file );
1482         owner.getClientManager().updateDocument( file.getPath(),
1483                                         ClientInfo.getPreferences().getProperty( ClientConstants.
1484                                         getName() ),
1485                                         rawFileContents,
1486                                         newRawFileContents );
1487         updateLocalDocument( file.getPath(), newRawFileContents );
1488     }
1489     catch ( IOException ioe )
1490     {
1491         System.err.println( ioe.toString() );
1492     }
1493
1494     }
1495     if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ) )
1496     {
1497         owner.advanceTask();
1498     }
1499     if ( getUpdateTimer().getDelay() > ( ClientConstants.getUPDATE_NOTIFICATION_THRESHOLD() * 1000 ) )
1500     {
1501         owner.resetMessageArea();
1502     }
1503 }
1504 /**
1505  * Apply difference array to a document in the panorama put there by the user of the client calling this
1506  * method.
1507  * Creation date: (22-08-00 01:56:16)
1508  * @param documentName java.lang.String
1509  * @param newRawFileContents byte[]
1510  */
1511 public synchronized void updateLocalDocument(String documentName, byte[] newRawFileContents)
1512 {
1513     String fqPath = ClientConstants.createFullyQualifiedDocumentName( ClientInfo.getPreferences().
1514                                         getProperty( ClientConstants.getName() ), documentName );
1515     ZVisualLeaf affectedNode = (ZVisualLeaf) getLocalDocuments().get( fqPath );
1516     PZSwing affectedVisualComponent = (PZSwing) affectedNode.getVisualComponent();
1517     JScrollPane affectedScrollPane = (JScrollPane) affectedVisualComponent.getComponent();
1518     if ( affectedScrollPane.getViewport().getView() instanceof JLabel )
1519     {
1520         affectedScrollPane.setViewport( new JLabel( new ImageIcon( newRawFileContents ) ) );
1521         affectedScrollPane.putClientProperty( ClientConstants.getRAW_BITMAP(), newRawFileContents );
1522     }
1523     else if ( affectedScrollPane.getViewport().getView() instanceof javax.swing.text.JTextComponent )
1524     {
1525         ( (javax.swing.text.JTextComponent) affectedScrollPane.getViewport().getView() ).setText( new
1526             String( newRawFileContents ) );
1527     }
1528     affectedScrollPane.setColumnHeaderView( newDocumentLabel(
1529                                         ClientConstants.getDocument_LABEL( (String) affectedNode.getClientProperty( ClientConstants.
1530                                         getDOCUMENT_NAME() ),

```

```

1526                               (String) affectedNode.getClientProperty( ClientConstants.getAUTHOR_NAME()
1527                                         ),
1528                                         newRawFileContents.length,
1529                                         new java.util.Date() )));
1530 }
1531 /**
1532 * Creation date: (23-06-00 21:44:36)
1533 */
1534 public void validateLayout()
1535 {
1536     getLayoutManager().doLayout( getAllDocumentsAsArray() );
1537     // System.out.println("validateLayout");
1538     revalidate();
1539     getDrawingSurface().repaint();
1540 }
1541 /**
1542 * Zooms and pans the panorama camera to the argument node so that it is centered in and scaled to the
1543 * viewport.
1544 * Creation date: (12-07-00 17:33:31)
1545 * @param node edu.umd.cs.jazz.ZNode
1546 */
1547 public void zoomAndPanToNode(ZVisualLeaf node)
1548 {
1549     java.awt.Rectangle targetRectangle = new java.awt.Rectangle( node.getVisualComponentGlobalBounds().
1550                                         getBounds() );
1551     PZoomHandler.setTargetRectangle( targetRectangle );
1552     getCamera().center( targetRectangle,
1553                         Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
1554                                         getANIMATION_DURATION_IN_MILLISECS() )),
1555                         getDrawingSurface() );
1556 }
1557 /**
1558 * Convenience method to allow calls using the documentID rather than the node itself
1559 * Creation date: (30-08-00 14:33:32)
1560 * @param documentID java.lang.String
1561 */
1562 public void zoomAndPanToNode(String documentID)
1563 {
1564     if ( getLocalDocuments().containsKey( documentID ) )
1565     {
1566         zoomAndPanToNode( (ZVisualLeaf) getLocalDocuments().get( documentID ) );
1567     }
1568     else
1569     {
1570         zoomAndPanToNode( (ZVisualLeaf) getDocuments().get( documentID ) );
1571     }
1572 }
1573 /**
1574 * Insert the method's description here.
1575 * Creation date: (28-07-00 01:15:23)
1576 */
1577 public void zoomDocument()
1578 {
1579     setZoomingNode( true );
1580     ( (PZSwing) currentlyClickedNode.getVisualComponent() ).setZoomingNode( currentlyClickedNode );
1581     boundsBeforeResizing = currentlyClickedNode.getVisualComponentBounds();
1582     previousMousePosition = null;
1583 }
1584 /**
1585 * Zoom and pan the camera so that the viewport contains an overview of all documents currently in the
1586 * panorama
1587 * Creation date: (01-08-00 15:23:15)
1588 */
1589 public void zoomToOverview()

```

```

1586 {
1587     if ( ClientInfo.getPreferences().getProperty( ClientConstants.getZOOM_TO_OVERVIEW_ON_UPDATE() ) .
1588         equalsIgnoreCase( ClientConstants.getZOOM_TO_OVERVIEW() ) )
1589     {
1590         if ( (getDocuments().size() + getLocalDocuments().size()) > 0 )
1591         {
1592             java.awt.Rectangle targetRectangle = computeOverviewRectangle( getLayer().getChildren() );
1593             PZoomHandler.setTargetRectangle( targetRectangle );
1594             getCamera().center( targetRectangle,
1595                 Integer.parseInt( ClientInfo.getPreferences().getProperty( ClientConstants.
1596                     getANIMATION_DURATION_IN_MILLISECS() ) ),
1597                 getDrawingSurface() );
1598             owner.getClientManager().clearTopicTree();
1599         }
1600     }
1601 }
1602 }
```

Listing C.8: ClientPanoramaLayoutManager.java

```

package clientapp;
2
4 import edu.umd.cs.jazz.*;
4 import edu.umd.cs.jazz.event.*;
5 import java.util.*;
6 /**
7  * The abstract root of the layout manager class hierarchy.
8  * Creation date: (12-10-00 12:36:38)
9  * @author:
10 */
11 public abstract class ClientPanoramaLayoutManager implements ZLayoutManager {
12     private final static java.lang.String DOCUMENT_WIDTH = "Document_width";
13     private final static java.lang.String DOCUMENT_HEIGHT = "Document_height";
14     private final static int DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
15     private final static int DOCUMENT_WIDTH_DEFAULT_VALUE = 500;
16 /**
17  * ClientPanoramaLayoutManager constructor comment.
18 */
19     public ClientPanoramaLayoutManager() {
20         super();
21     }
22 /**
23  * Insert the method's description here.
24  * Creation date: (12-10-00 13:06:25)
25  * @return java.lang.Object
26 */
27     public Object clone() {
28         return null;
29     }
30 /**
31  * Insert the method's description here.
32  * Creation date: (12-10-00 23:47:58)
33  * @param nodes edu.umd.cs.jazz.ZVisualLeaf[]
34 */
35     public void doLayout(ZNode[] nodes)
36     {
37     }
38 /**
39  * doLayout method comment.
40 */
41     public void doLayout(ZGroup node) {}
42 /**
43  * Insert the method's description here.
44  * Creation date: (12-10-00 14:02:19)
45  * @return java.lang.String
46 */
47 }
```

```

    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 */
55 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 */
63 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 */
71 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 */
79 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
 */
85 public boolean nodeLocked(ZNode node)
{
86     if ( node.getClientProperty( ClientConstants.getNODE_LOCKED() )!= null )
87     {
88         return node.getClientProperty( ClientConstants.getNODE_LOCKED() ).equals( ClientConstants.getTRUE
89             () )? true : false;
90     }
91     return false;
}
92 /**
 * postLayout method comment.
93 */
94 public void postLayout(ZGroup node) {}
95 /**
 * preLayout method comment.
96 */
97 public void preLayout(ZGroup node) {}
98 /**
 * Non-overridable method to position individual node. Any subclasses should use this method rather than
99     roll their own
100    * in order to hide non-essential panorama implementation detail.
101    * Creation date: (14-10-00 13:19:05)
102    * @param node edu.umd.cs.jazz.ZNode
103    * @param xCoordinate float
104    * @param yCoordinate float
 */
105 public final boolean setPosition(ZNode node, float xCoordinate, float yCoordinate)

```

```

112     {
113         // System.out.println( node.getClientProperty( ClientConstants.getNODE_LOCKED() ) );
114         if ( !nodeLocked( node )== false )
115         {
116             node.editor().getTransformGroup().setTranslation( xCoordinate, yCoordinate );
117             return true;
118         }
119         else
120         {
121             return false;
122         }
123     }
124 /**
125 * Insert the method's description here.
126 * Creation date: (14-10-00 13:23:49)
127 * @param node edu.umd.cs.jazz.ZNode
128 * @param position java.awt.Dimension
129 */
130 public final boolean setPosition(ZNode node, java.awt.Point position)
131 {
132     return setPosition( node, (float) position.getX(), (float) position.getY() );
133 }
134 /**
135 * The newSettings argument is a set of manager specific <key,value> pairs that define the values to be
136 * displayed in the
137 * customization box.
138 * Creation date: (12-10-00 13:13:06)
139 * @param newSettings java.util.Properties
140 */
141 public void setSettings(java.util.Properties newSettings) {
142 }
143 /**
144 * setState method comment.
145 */
146 public void setState(String fieldType, String fieldName, Object fieldValue) {}
147 /**
148 * Insert the method's description here.
149 * Creation date: (12-10-00 12:58:42)
150 */
151 public void show()
152 {
153     show( null );
154 }
155 /**
156 * Insert the method's description here.
157 * Creation date: (12-10-00 12:56:45)
158 * @param owner javax.swing.JDialog
159 */
160 public void show(javax.swing.JDialog owner) {}
161 /**
162 * Insert the method's description here.
163 * Creation date: (12-10-00 15:04:17)
164 * @return java.util.Properties
165 * @param currentSettings java.util.Properties
166 */
167 public abstract Properties updateSettings(Properties currentSettings);
168 /**
169 * writeObject method comment.
170 */
171 public void writeObject(edu.umd.cs.jazz.io.ZObjectOutputStream out) throws java.io.IOException {}
172 /**
173 * writeObjectRecurse method comment.
174 */
175 public void writeObjectRecurse(edu.umd.cs.jazz.io.ZObjectOutputStream out) throws java.io.IOException
176 {}

```

176 }

Listing C.9: ComposeNewMessage.java

```
package clientapp;
2
4 import javax.swing.event.*;
5 import java.awt.event.*;
6 import peerviewmisc.*;
7 import javax.swing.JMenuItem;
8 import java.awt.datatransfer.*;
9
10 /**
11 * This class implements the dialog box for composing new messages
12 * Creation date: (10-07-00 23:54:05)
13 * @author:
14 */
15 public class ComposeNewMessage extends javax.swing.JDialog
16 {
17     public class PopUpActionListener implements ActionListener
18     {
19         public void actionPerformed( ActionEvent ae )
20         {
21             if ( ae.getSource() instanceof JMenuItem )
22             {
23                 JMenuItem menuItem = (JMenuItem) ae.getSource();
24
25                 if ( menuItem == getPasteTextItem() )
26                 {
27                     pasteText();
28                 }
29             }
30         }
31     };
32
33     public class MouseHandler extends MouseAdapter
34     {
35         public void mouseClicked( MouseEvent me )
36         {
37             // if right-click
38             if ( me.isMetaDown() )
39             {
40                 popUpPopUpMenu( me.getPoint() );
41             }
42         }
43         public void mouseEntered( MouseEvent me )
44         {
45             popDownPopUpMenu();
46         }
47     };
48
49     public class PopUpMouseHandler extends MouseAdapter
50     {
51         public void mouseExited( MouseEvent me )
52         {
53             popDownPopUpMenu();
54         }
55
56         public void mouseEntered( MouseEvent me )
57         {
58             popDownPopUpMenu();
59         }
60     };
61
62     public class BoxMouseHandler extends MouseMotionAdapter
63     {
```

```

64     public void mouseMoved( MouseEvent me )
65     {
66         popDownPopUpMenu();
67     }
68 }

70     public class SubmitButtonHandler implements ActionListener
71     {
72         public void actionPerformed( ActionEvent ae )
73         {
74             submitAndClose();
75         }
76     }

78     public class DiscardButtonHandler implements ActionListener
79     {
80         public void actionPerformed( ActionEvent ae )
81         {
82             discardAndClose();
83         }
84     }

85     private javax.swing.JLabel ivjAuthorLabel = null;
86     private javax.swing.JLabel ivjDateLabel = null;
87     private javax.swing.JButton ivjDiscardButton = null;
88     private javax.swing.JPanel ivjJDialogContentPane = null;
89     private javax.swing.JButton ivjSubmitButton = null;
90     private javax.swing.JTextField ivjAuthorTextField = null;
91     private javax.swing.JEditorPane ivjContentsField = null;
92     private javax.swing.JTextField ivjDateField = null;
93     private javax.swing.JPanel ivjButtonPanel = null;
94     private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
95     private javax.swing.JScrollPane ivjContentsPanel = null;
96     private javax.swing.JPanel ivjHeaderPanel = null;
97     private javax.swing.JTextField ivjTitleField = null;
98     private javax.swing.JLabel ivjTitleLabel = null;
99     private java.util.Date date = null;
100    private peerviewmisc.Message message;
101    private javax.swing.JMenuItem ivjPasteTextItem = null;
102    private javax.swing.JPopupMenu ivjPopupMenu = null;
103    private ClientApplication clientApplication = null;

104 /**
105  * ComposeNewMessage constructor comment.
106  */
107 public ComposeNewMessage() {
108     super();
109     initialize();
110 }

111 /**
112  * ComposeNewMessage constructor comment.
113  * @param owner java.awt.Dialog
114  */
115 public ComposeNewMessage(java.awt.Dialog owner) {
116     super(owner);
117 }

118 /**
119  * ComposeNewMessage constructor comment.
120  * @param owner java.awt.Dialog
121  * @param title java.lang.String
122  */
123 public ComposeNewMessage(java.awt.Dialog owner, String title) {
124     super(owner, title);
125 }

126 /**
127  * ComposeNewMessage constructor comment.
128  * @param owner java.awt.Dialog
129  * @param title java.lang.String

```

```

130 * @param modal boolean
131 */
132 public ComposeNewMessage(java.awt.Dialog owner, String title, boolean modal) {
133     super(owner, title, modal);
134 }
135 /**
136 * ComposeNewMessage constructor comment.
137 * @param owner java.awt.Dialog
138 * @param modal boolean
139 */
140 public ComposeNewMessage(java.awt.Dialog owner, boolean modal) {
141     super(owner, modal);
142 }
143 /**
144 * ComposeNewMessage constructor comment.
145 * @param owner java.awt.Frame
146 */
147 public ComposeNewMessage(java.awt.Frame owner) {
148     super(owner);
149 }
150 /**
151 * ComposeNewMessage constructor comment.
152 * @param owner java.awt.Frame
153 * @param title java.lang.String
154 */
155 public ComposeNewMessage(java.awt.Frame owner, String title) {
156     super(owner, title);
157 }
158 /**
159 * ComposeNewMessage constructor comment.
160 * @param owner java.awt.Frame
161 * @param title java.lang.String
162 * @param modal boolean
163 */
164 public ComposeNewMessage(java.awt.Frame owner, String title, boolean modal) {
165     super(owner, title, modal);
166 }
167 /**
168 * ComposeNewMessage constructor comment.
169 * @param owner java.awt.Frame
170 * @param modal boolean
171 */
172 public ComposeNewMessage(java.awt.Frame owner, boolean modal) {
173     super(owner, modal);
174 }
175 /**
176 * Insert the method's description here.
177 * Creation date: (11-07-00 01:12:03)
178 */
179 public void discardAndClose()
180 {
181     message = null;
182     dispose();
183 }
184 /**
185 * Insert the method's description here.
186 * Creation date: (11-07-00 01:19:45)
187 * @return java.lang.String
188 */
189 public String getAuthor() {
190     return getAuthorTextField().getText();
191 }
192 /**
193 * Return the AuthorLabel property value.
194 * @return javax.swing.JLabel
195 */

```

```

196 /* WARNING: THIS METHOD WILL BE REGENERATED. */
197 private javax.swing.JLabel getAuthorLabel() {
198     if (ivjAuthorLabel == null) {
199         try {
200             ivjAuthorLabel = new javax.swing.JLabel();
201             ivjAuthorLabel.setName("AuthorLabel");
202             ivjAuthorLabel.setText("Author");
203             // user code begin {1}
204             // user code end
205         } catch (java.lang.Throwable ivjExc) {
206             // user code begin {2}
207             // user code end
208             handleException(ivjExc);
209         }
210     }
211     return ivjAuthorLabel;
212 }
213 /**
214 * Return the JTextField1 property value.
215 * @return javax.swing.JTextField
216 */
217 /* WARNING: THIS METHOD WILL BE REGENERATED. */
218 private javax.swing.JTextField getAuthorTextField() {
219     if (ivjAuthorTextField == null) {
220         try {
221             ivjAuthorTextField = new javax.swing.JTextField();
222             ivjAuthorTextField.setName("AuthorTextField");
223             // user code begin {1}
224             ivjAuthorTextField.setEditable( false );
225             // user code end
226         } catch (java.lang.Throwable ivjExc) {
227             // user code begin {2}
228             // user code end
229             handleException(ivjExc);
230         }
231     }
232     return ivjAuthorTextField;
233 }
234 /**
235 * Return the JPanel2 property value.
236 * @return javax.swing.JPanel
237 */
238 /* WARNING: THIS METHOD WILL BE REGENERATED. */
239 private javax.swing.JPanel getButtonPanel() {
240     if (ivjButtonPanel == null) {
241         try {
242             ivjButtonPanel = new javax.swing.JPanel();
243             ivjButtonPanel.setName("ButtonPanel");
244             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
245             getButtonPanel().add(getSubmitButton(), getSubmitButton().getName());
246             getButtonPanel().add(getDiscardButton(), getDiscardButton().getName());
247             // user code begin {1}
248             // user code end
249         } catch (java.lang.Throwable ivjExc) {
250             // user code begin {2}
251             // user code end
252             handleException(ivjExc);
253         }
254     }
255     return ivjButtonPanel;
256 }
257 /**
258 * Return the ButtonPanelFlowLayout property value.
259 * @return java.awt.FlowLayout
260 */
261 /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

262 private java.awt.FlowLayout getButtonPanelFlowLayout() {
263     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
264     try {
265         /* Create part */
266         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
267         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
268     } catch (java.lang.Throwable ivjExc) {
269         handleException(ivjExc);
270     };
271     return ivjButtonPanelFlowLayout;
272 }
273 /**
274 * Insert the method's description here.
275 * Creation date: (07-08-00 23:35:48)
276 * @return clientapp.ClientApplication
277 */
278 public ClientApplication getClientApplication() {
279     return clientApplication;
280 }
281 /**
282 * Insert the method's description here.
283 * Creation date: (11-07-00 01:19:58)
284 * @return java.lang.String
285 */
286 public String getContents() {
287     return getContentsField().getText();
288 }
289 /**
290 * Return the JEditorPane1 property value.
291 * @return javax.swing.JEditorPane
292 */
293 /* WARNING: THIS METHOD WILL BE REGENERATED. */
294 private javax.swing.JEditorPane getContentsField() {
295     if (ivjContentsField == null) {
296         try {
297             ivjContentsField = new javax.swing.JEditorPane();
298             ivjContentsField.setName("ContentsField");
299             ivjContentsField.setBounds(0, 0, 6, 23);
300             // user code begin {1}
301             ivjContentsField.addMouseListener( new MouseHandler() );
302             // user code end
303         } catch (java.lang.Throwable ivjExc) {
304             // user code begin {2}
305             // user code end
306             handleException(ivjExc);
307         }
308     }
309     return ivjContentsField;
310 }
311 /**
312 * Return the JScrollPane1 property value.
313 * @return javax.swing.JScrollPane
314 */
315 /* WARNING: THIS METHOD WILL BE REGENERATED. */
316 private javax.swing.JScrollPane getContentsPanel() {
317     if (ivjContentsPanel == null) {
318         try {
319             ivjContentsPanel = new javax.swing.JScrollPane();
320             ivjContentsPanel.setName("ContentsPanel");
321             getContentsPanel().setViewportView(getContentsField());
322             // user code begin {1}
323             getContentsPanel().setBorder( javax.swing.BorderFactory.createMatteBorder(
324                 ClientConstants.getBORDER_WIDTH(),
325                 ClientConstants.getBORDER_WIDTH(),
326                 ClientConstants.getBORDER_WIDTH(),

```

```

328         // user code end
329     } catch (java.lang.Throwable ivjExc) {
330         // user code begin {2}
331         // user code end
332         handleException(ivjExc);
333     }
334 }
335 return ivjContentsPanel;
336 }
337 /**
338 * Insert the method's description here.
339 * Creation date: (16-07-00 10:56:47)
340 * @return java.util.GregorianCalendar
341 */
342 public java.util.Date getDate() {
343     return date;
344 }
345 /**
346 * Return the JTextField2 property value.
347 * @return javax.swing.JTextField
348 */
349 /* WARNING: THIS METHOD WILL BE REGENERATED. */
350 private javax.swing.JTextField getDateField() {
351     if (ivjDateField == null) {
352         try {
353             ivjDateField = new javax.swing.JTextField();
354             ivjDateField.setName("DateField");
355             // user code begin {1}
356             // set date field text string to current time as obtained by instantiating Date and converting
357             // it to string
358             date = new java.util.Date();
359             ivjDateField.setText( date.toString() );
360             ivjDateField.setEditable( false );
361             // user code end
362         } catch (java.lang.Throwable ivjExc) {
363             // user code begin {2}
364             // user code end
365             handleException(ivjExc);
366         }
367     }
368     return ivjDateField;
369 }
370 /**
371 * Return the DateLabel property value.
372 * @return javax.swing.JLabel
373 */
374 /* WARNING: THIS METHOD WILL BE REGENERATED. */
375 private javax.swing.JLabel getDateLabel() {
376     if (ivjDateLabel == null) {
377         try {
378             ivjDateLabel = new javax.swing.JLabel();
379             ivjDateLabel.setName("DateLabel");
380             ivjDateLabel.setText("Date");
381             // user code begin {1}
382             // user code end
383         } catch (java.lang.Throwable ivjExc) {
384             // user code begin {2}
385             // user code end
386             handleException(ivjExc);
387         }
388     }
389     return ivjDateLabel;
390 }
391 /**
392 * Return the DiscardButton property value.

```

```

392 * @return javax.swing.JButton
393 */
394 /* WARNING: THIS METHOD WILL BE REGENERATED. */
395 private javax.swing.JButton getDiscardButton() {
396     if (ivjDiscardButton == null) {
397         try {
398             ivjDiscardButton = new javax.swing.JButton();
399             ivjDiscardButton.setName("DiscardButton");
400             ivjDiscardButton.setMnemonic('D');
401             ivjDiscardButton.setText("Discard");
402             // user code begin {1}
403             ivjDiscardButton.addActionListener( new DiscardButtonHandler() );
404             // user code end
405         } catch (java.lang.Throwable ivjExc) {
406             // user code begin {2}
407             // user code end
408             handleException(ivjExc);
409         }
410     }
411     return ivjDiscardButton;
412 }
413 /**
414 * Return the JPanel1 property value.
415 * @return javax.swing.JPanel
416 */
417 /* WARNING: THIS METHOD WILL BE REGENERATED. */
418 private javax.swing.JPanel getHeaderPanel() {
419     if (ivjHeaderPanel == null) {
420         try {
421             ivjHeaderPanel = new javax.swing.JPanel();
422             ivjHeaderPanel.setName("HeaderPanel");
423             ivjHeaderPanel.setLayout(new java.awt.GridBagLayout());
424
425             java.awt.GridBagConstraints constraintsAuthorLabel = new java.awt.GridBagConstraints();
426             constraintsAuthorLabel.gridx = 0; constraintsAuthorLabel.gridy = 0;
427             constraintsAuthorLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
428             constraintsAuthorLabel.insets = new java.awt.Insets(4, 4, 4, 4);
429             getHeaderPanel().add(getAuthorLabel(), constraintsAuthorLabel);
430
431             java.awt.GridBagConstraints constraintsDateLabel = new java.awt.GridBagConstraints();
432             constraintsDateLabel.gridx = 0; constraintsDateLabel.gridy = 1;
433             constraintsDateLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
434             constraintsDateLabel.insets = new java.awt.Insets(4, 4, 4, 4);
435             getHeaderPanel().add(getDateLabel(), constraintsDateLabel);
436
437             java.awt.GridBagConstraints constraintsTitleLabel = new java.awt.GridBagConstraints();
438             constraintsTitleLabel.gridx = 0; constraintsTitleLabel.gridy = 2;
439             constraintsTitleLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
440             constraintsTitleLabel.insets = new java.awt.Insets(4, 4, 4, 4);
441             getHeaderPanel().add getTitleLabel(), constraintsTitleLabel);
442
443             java.awt.GridBagConstraints constraintsAuthorTextField = new java.awt.GridBagConstraints();
444             constraintsAuthorTextField.gridx = 1; constraintsAuthorTextField.gridy = 0;
445             constraintsAuthorTextField.fill = java.awt.GridBagConstraints.BOTH;
446             constraintsAuthorTextField.anchor = java.awt.GridBagConstraints.EAST;
447             constraintsAuthorTextField.weightx = 1.0;
448             constraintsAuthorTextField.ipadx = 1;
449             constraintsAuthorTextField.insets = new java.awt.Insets(4, 4, 4, 4);
450             getHeaderPanel().add(getAuthorTextField(), constraintsAuthorTextField);
451
452             java.awt.GridBagConstraints constraintsDateField = new java.awt.GridBagConstraints();
453             constraintsDateField.gridx = 1; constraintsDateField.gridy = 1;
454             constraintsDateField.fill = java.awt.GridBagConstraints.HORIZONTAL;
455             constraintsDateField.anchor = java.awt.GridBagConstraints.EAST;
456             constraintsDateField.weightx = 1.0;
457             constraintsDateField.insets = new java.awt.Insets(4, 4, 4, 4);

```

```
458     getHeaderPanel().add(getDateField(), constraintsDateField);
459
460     java.awt.GridBagConstraints constraintsTitleField = new java.awt.GridBagConstraints();
461     constraintsTitleField.gridx = 1; constraintsTitleField.gridy = 2;
462     constraintsTitleField.fill = java.awt.GridBagConstraints.HORIZONTAL;
463     constraintsTitleField.weightx = 1.0;
464     constraintsTitleField.insets = new java.awt.Insets(4, 4, 4, 4);
465     getHeaderPanel().add getTitleField(), constraintsTitleField);
466 // user code begin {1}
467     getHeaderPanel().setBorder( javax.swing.BorderFactory.createMatteBorder( ClientConstants.
468         getBORDER_WIDTH(),
469
470                                         ClientConstants.getBORDER_WIDTH(),
471                                         ClientConstants.getBORDER_WIDTH(),
472                                         ClientConstants.getBORDER_WIDTH(),
473                                         getHeaderPanel().getBackground() ) );
474
475     // user code end
476 } catch (java.lang.Throwable ivjExc) {
477     // user code begin {2}
478     // user code end
479     handleException(ivjExc);
480 }
481
482     return ivjHeaderPanel;
483 }
484 /**
485 * Return the JDialoContentPane property value.
486 * @return javax.swing.JPanel
487 */
488 /* WARNING: THIS METHOD WILL BE REGENERATED. */
489 private javax.swing.JPanel getJDialoContentPane() {
490     if (ivjJDialoContentPane == null) {
491         try {
492             ivjJDialoContentPane = new javax.swing.JPanel();
493             ivjJDialoContentPane.setName("JDialoContentPane");
494             ivjJDialoContentPane.setLayout(new java.awt.BorderLayout());
495             getJDialoContentPane().add(getHeaderPanel(), "North");
496             getJDialoContentPane().add(getContentsPanel(), "Center");
497             getJDialoContentPane().add(getButtonPanel(), "South");
498             // user code begin {1}
499             // user code end
500         } catch (java.lang.Throwable ivjExc) {
501             // user code begin {2}
502             // user code end
503             handleException(ivjExc);
504         }
505     }
506     return ivjJDialoContentPane;
507 }
508 /**
509 * Insert the method's description here.
510 * Creation date: (18-07-00 08:37:59)
511 * @return peerviewmisc.Message
512 */
513 public peerviewmisc.Message getMessage() {
514     return message;
515 }
516 /**
517 * Return the PasteTextItem property value.
518 * @return javax.swing.JMenuItem
519 */
520 /* WARNING: THIS METHOD WILL BE REGENERATED. */
521 private javax.swing.JMenuItem getPasteTextItem() {
522     if (ivjPasteTextItem == null) {
523         try {
524             ivjPasteTextItem = new javax.swing.JMenuItem();
525             ivjPasteTextItem.setName("PasteTextItem");
526         }
527     }
528 }
```

```

524         ivjPasteTextItem.setMnemonic('P');
525         ivjPasteTextItem.setText("Paste_Utext");
526         // user code begin {1}
527         ivjPasteTextItem.addActionListener( new PopUpActionListener() );
528         // user code end
529     } catch (java.lang.Throwable ivjExc) {
530         // user code begin {2}
531         // user code end
532         handleException(ivjExc);
533     }
534     return ivjPasteTextItem;
535 }
536 /**
537 * Return the PopUpMenu property value.
538 * @return javax.swing.JPopupMenu
539 */
540 /* WARNING: THIS METHOD WILL BE REGENERATED. */
541 private javax.swing.JPopupMenu getPopUpMenu() {
542     if (ivjPopUpMenu == null) {
543         try {
544             ivjPopUpMenu = new javax.swing.JPopupMenu();
545             ivjPopUpMenu.setName("PopUpMenu");
546             ivjPopUpMenu.add(getPasteTextItem());
547             // user code begin {1}
548             // user code end
549         } catch (java.lang.Throwable ivjExc) {
550             // user code begin {2}
551             // user code end
552             handleException(ivjExc);
553         }
554     }
555     return ivjPopUpMenu;
556 }
557 /**
558 * Return the SubmitButton property value.
559 * @return javax.swing.JButton
560 */
561 /* WARNING: THIS METHOD WILL BE REGENERATED. */
562 private javax.swing.JButton getSubmitButton() {
563     if (ivjSubmitButton == null) {
564         try {
565             ivjSubmitButton = new javax.swing.JButton();
566             ivjSubmitButton.setName("SubmitButton");
567             ivjSubmitButton.setMnemonic('s');
568             ivjSubmitButton.setText("Submit");
569             // user code begin {1}
570             ivjSubmitButton.addActionListener( new SubmitButtonHandler() );
571             // user code end
572         } catch (java.lang.Throwable ivjExc) {
573             // user code begin {2}
574             // user code end
575             handleException(ivjExc);
576         }
577     }
578     return ivjSubmitButton;
579 }
580 /**
581 * Insert the method's description here.
582 * Creation date: (11-07-00 01:25:12)
583 * @return java.lang.String
584 */
585 public String getTitle() {
586     return getTitleField().getText();
587 }
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) {
595         try {
596             ivjTitleField = new javax.swing.JTextField();
597             ivjTitleField.setName("TitleField");
598             // user code begin {1}
599             // user code end
600         } catch (java.lang.Throwable ivjExc) {
601             // user code begin {2}
602             // user code end
603             handleException(ivjExc);
604         }
605     }
606     return ivjTitleField;
607 }
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) {
615         try {
616             ivjTitleLabel = new javax.swing.JLabel();
617             ivjTitleLabel.setName("TitleLabel");
618             ivjTitleLabel.setText("Title");
619             // user code begin {1}
620             // user code end
621         } catch (java.lang.Throwable ivjExc) {
622             // user code begin {2}
623             // user code end
624             handleException(ivjExc);
625         }
626     }
627     return ivjTitleLabel;
628 }
629 /**
630 * Called whenever the part throws an exception.
631 * @param exception java.lang.Throwable
632 */
private void handleException(java.lang.Throwable exception) {
633     /* Uncomment the following lines to print uncaught exceptions to stdout */
634     // System.out.println("----- UNCAUGHT EXCEPTION -----");
635     // exception.printStackTrace(System.out);
636 }
637 /**
638 * Initialize the class.
639 */
640 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
641     try {
642         // user code begin {1}
643         // user code end
644         setName("ComposeNewMessage");
645         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
646         setSize(633, 490);
647         setModal(true);
648         setTitle("Compose\u00a5new\u00a5message");
649         setContentPane(getJDialogContentPane());
650     } catch (java.lang.Throwable ivjExc) {
651         handleException(ivjExc);
652     }
653 }

```

```

    }
656 // user code begin {2}
657 // default positioning puts the dialog at center of screen. The method used is a slight hack, but
658 // works fine.
659 setLocationRelativeTo( getOwner() );
660 // user code end
661 }
662 /**
663 * main entrypoint - starts the part when it is run as an application
664 * @param args java.lang.String[]
665 */
666 public static void main(java.lang.String[] args) {
667 try {
668     ComposeNewMessage aComposeNewMessage;
669     aComposeNewMessage = new ComposeNewMessage();
670     aComposeNewMessage.setModal(true);
671     aComposeNewMessage.addWindowListener(new java.awt.event.WindowAdapter() {
672         public void windowClosing(java.awt.event.WindowEvent e) {
673             System.exit(0);
674         };
675     });
676     aComposeNewMessage.setVisible(true);
677 } catch (Throwable exception) {
678     System.err.println("Exception occurred in main() of javax.swing.JDialog");
679     exception.printStackTrace(System.out);
680 }
681 /**
682 * Paste the contents of the clipboard, if any, into the message at the current caret position.
683 * Creation date: (07-08-00 23:31:17)
684 */
685 public void pasteText()
686 {
687     getContentsField().paste();
688 }
689 /**
690 * Insert the method's description here.
691 * Creation date: (08-08-00 10:03:32)
692 */
693 public void popDownPopupMenu()
694 {
695     if ( getPopupMenu().isVisible() )
696     {
697         getPopupMenu().setVisible( false );
698     }
699 }
700 /**
701 * Display the pop-up menu.
702 * Creation date: (07-08-00 23:20:22)
703 */
704 public void popUpPopupMenu( java.awt.Point position )
705 {
706     getPopupMenu().show( getContentsPanel(), (int) position.getX(), (int) position.getY() );
707 }
708 /**
709 * Insert the method's description here.
710 * Creation date: (11-07-00 00:38:35)
711 * @param newAuthorName java.lang.String
712 */
713 public void setAuthor(String newAuthorName)
714 {
715     getAuthorTextField().setText( newAuthorName );
716 }
717 /**
718 * Insert the method's description here.
719 * Creation date: (07-08-00 23:35:48)

```

```

720 * @param newClientApplication clientapp.ClientApplication
721 */
722 public void setClientApplication(ClientApplication newClientApplication) {
723     clientApplication = newClientApplication;
724 }
725 /**
726 * Insert the method's description here.
727 * Creation date: (16-07-00 10:56:47)
728 * @param newDate java.util.GregorianCalendar
729 */
730 public void setDate(java.util.Date newDate) {
731     date = newDate;
732 }
733 /**
734 * Insert the method's description here.
735 * Creation date: (18-07-00 08:37:59)
736 * @param newMessage peerviewmisc.Message
737 */
738 public void setMessage(peerviewmisc.Message newMessage) {
739     message = newMessage;
740 }
741 /**
742 * Package the contents of the message box fields into a message data structure and then dispose of the
743 * box itself.
744 * The message structure is part of the dialog box object and can be retrieved by the caller after the
745 * box is disposed of.
746 * Creation date: (11-07-00 01:11:45)
747 */
748 public void submitAndClose()
749 {
750     if ( getTitleField().getText().length() == 0 )
751     {
752         MessageBox messageBox = new MessageBox();
753         messageBox.setText( ClientConstants.getTITLE_REQUIRED_MESSAGE() );
754         messageBox.show();
755         return;
756     }
757     message = new peerviewmisc.Message();
758     message.setAuthor( getAuthor() );
759     message.setContents( getContents() + ClientConstants.getCONTENTS_AND_SIGNATURE_DIVIDER() + ClientInfo
760                         .getPreferences().getProperty( ClientConstants.getSIGNATURE() ) );
761     message.setDate( getDate() );
762     message.setTitle( getTitle() );
763     dispose();
764 }

```

Listing C.10: DiscussionFrame.java

```

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

```

```

18     private JMenuBar ivjDiscussionFrameJMenuBar = null;
19     private JMenu ivjDiscussionMenu = null;
20     private JTree ivjDiscussionTree = null;
21     private JMenu ivjHelpMenu = null;
22     private JPanel ivjJFrameContentPane = null;
23     private JPanel ivj JPanel1 = null;
24     private JPanel ivj JPanel2 = null;
25     private JSeparator ivj JSeparator1 = null;
26     private JSplitPane ivj JSplitPanel1 = null;
27     private JToolBar ivj JToolBar1 = null;
28     private JButton ivj JToolBarButton1 = null;
29     private JButton ivj JToolBarButton2 = null;
30     private JButton ivj JToolBarButton3 = null;
31     private JTextPane ivj MessageDisplayArea = null;
32     private JMenuItem ivj NewMessageItem = null;
33     private JMenuItem ivj NewThreadItem = null;
34     private JLabel ivj Author = null;
35     private JLabel ivj AuthorLabel = null;
36     private JLabel ivj DateLabel = null;
37     private JLabel ivj JLabel1 = null;
38     private JLabel ivj JLabel2 = null;
39     private BoxLayout ivj JPanel2BoxLayout = null;
40     private JPanel ivj JPanelHoldingHeader = null;
41     private JPanel ivj JPanelHoldingHeaderFields = null;
42     private JPanel ivj JPanelHoldingHeadingLabels = null;
43     private JLabel ivj SubjectLabel = null;
44     private BoxLayout ivj JPanelHoldingHeaderFieldsBoxLayout = null;
45     private BoxLayout ivj JPanelHoldingHeadingLabelsBoxLayout = null;
46     private JLabel ivj AuthorLabel1 = null;
47     private JDialo g ivj ComposeContribution = null;
48     private JLabel ivj DateLabel1 = null;
49     private JPanel ivj JDialo gContentPane = null;
50     private JPanel ivj MessagePanel = null;
51     private BoxLayout ivj MessagePanelBoxLayout = null;
52     private JLabel ivj SubjectLabel1 = null;
53     IvjEventHandler ivj EventHandler = new IvjEventHandler();
54     private JTextPane ivj MessageDisplayAreaOnComposeBox = null;
55     private JPanel ivj JPanelHoldingHeaderFieldsOnComposeBox = null;
56     private BoxLayout ivj JPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = null;
57     private JPanel ivj JPanelHoldingHeaderOnComposeBox = null;
58     private JPanel ivj JPanelHoldingHeadingLabelsOnComposeBox = null;
59     private BoxLayout ivj JPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = null;
60     private JPanel ivj JPanel3 = null;
61     private FlowLayout ivj JPanel3FlowLayout = null;
62     private BoxLayout ivj JDialo gContentPaneBoxLayout = null;
63     private JTextField ivj JTextField1 = null;
64     private JTextField ivj JTextField11 = null;
65     private JTextField ivj JTextField12 = null;
66     private JPanel ivj JPanel4 = null;
67     private JButton ivj DiscardButton = null;
68     private JButton ivj SubmitButton = null;

69     class IvjEventHandler implements java.awt.event.ActionListener {
70         public void actionPerformed(java.awt.event.ActionEvent e) {
71             if (e.getSource() == DiscussionFrame.this.getNewMessageItem())
72                 connEtoM1(e);
73             if (e.getSource() == DiscussionFrame.this.getJToolBarButton1())
74                 connEtoM2(e);
75         };
76     };
77 }
78 /**
79 * DiscussionFrame constructor comment.
80 */
81 public DiscussionFrame() {
82     super();
83     initialize();

```

```

84 }
85 /**
86 * DiscussionFrame constructor comment.
87 * @param title java.lang.String
88 */
89 public DiscussionFrame(String title) {
90     super(title);
91 }
92 /**
93 * connEtoM1: (NewMessageItem.action.actionPerformed(java.awt.event.ActionEvent) --> ComposeContribution
94 * .show()V)
95 * @param arg1 java.awt.event.ActionEvent
96 */
97 /* WARNING: THIS METHOD WILL BE REGENERATED. */
98 private void connEtoM1(java.awt.event.ActionEvent arg1) {
99     try {
100         // user code begin {1}
101         // user code end
102         getComposeContribution().show();
103         // user code begin {2}
104         // user code end
105     } catch (java.lang.Throwable ivjExc) {
106         // user code begin {3}
107         // user code end
108         handleException(ivjExc);
109     }
110 }
111 /**
112 * connEtoM2: (JToolBarButton1.action.actionPerformed(java.awt.event.ActionEvent) -->
113 * ComposeContribution.show()V)
114 * @param arg1 java.awt.event.ActionEvent
115 */
116 /* WARNING: THIS METHOD WILL BE REGENERATED. */
117 private void connEtoM2(java.awt.event.ActionEvent arg1) {
118     try {
119         // user code begin {1}
120         // user code end
121         getComposeContribution().show();
122         // user code begin {2}
123         // user code end
124     } catch (java.lang.Throwable ivjExc) {
125         // user code begin {3}
126         // user code end
127         handleException(ivjExc);
128     }
129 }
130 /**
131 * Return the AboutItem property value.
132 * @return javax.swing.JMenuItem
133 */
134 /* WARNING: THIS METHOD WILL BE REGENERATED. */
135 private javax.swing.JMenuItem getAboutItem() {
136     if (ivjAboutItem == null) {
137         try {
138             ivjAboutItem = new javax.swing.JMenuItem();
139             ivjAboutItem.setName("AboutItem");
140             ivjAboutItem.setText("About");
141             // user code begin {1}
142             // user code end
143         } catch (java.lang.Throwable ivjExc) {
144             // user code begin {2}
145             // user code end
146             handleException(ivjExc);
147         }
148     }
149     return ivjAboutItem;

```

```

148 }
149 /**
150 * Return the Author property value.
151 * @return javax.swing.JLabel
152 */
153 /* WARNING: THIS METHOD WILL BE REGENERATED. */
154 private javax.swing.JLabel getAuthor() {
155     if (ivjAuthor == null) {
156         try {
157             ivjAuthor = new javax.swing.JLabel();
158             ivjAuthor.setName("Author");
159             ivjAuthor.setText("JLabel1");
160             ivjAuthor.setForeground(java.awt.SystemColor.controlText);
161             ivjAuthor.setMinimumSize(new java.awt.Dimension(45, 23));
162             ivjAuthor.setMaximumSize(new java.awt.Dimension(450, 23));
163             // user code begin {1}
164             // user code end
165         } catch (java.lang.Throwable ivjExc) {
166             // user code begin {2}
167             // user code end
168             handleException(ivjExc);
169         }
170     }
171     return ivjAuthor;
172 }
173 /**
174 * Return the AuthorLabel property value.
175 * @return javax.swing.JLabel
176 */
177 /* WARNING: THIS METHOD WILL BE REGENERATED. */
178 private javax.swing.JLabel getAuthorLabel() {
179     if (ivjAuthorLabel == null) {
180         try {
181             ivjAuthorLabel = new javax.swing.JLabel();
182             ivjAuthorLabel.setName("AuthorLabel");
183             ivjAuthorLabel.setText("Author:");
184             ivjAuthorLabel.setMaximumSize(new java.awt.Dimension(41, 23));
185             ivjAuthorLabel.setMinimumSize(new java.awt.Dimension(41, 23));
186             // user code begin {1}
187             // user code end
188         } catch (java.lang.Throwable ivjExc) {
189             // user code begin {2}
190             // user code end
191             handleException(ivjExc);
192         }
193     }
194     return ivjAuthorLabel;
195 }
196 /**
197 * Return the AuthorLabel1 property value.
198 * @return javax.swing.JLabel
199 */
200 /* WARNING: THIS METHOD WILL BE REGENERATED. */
201 private javax.swing.JLabel getAuthorLabel1() {
202     if (ivjAuthorLabel1 == null) {
203         try {
204             ivjAuthorLabel1 = new javax.swing.JLabel();
205             ivjAuthorLabel1.setName("AuthorLabel1");
206             ivjAuthorLabel1.setPreferredSize(new java.awt.Dimension(41, 23));
207             ivjAuthorLabel1.setText("Author:");
208             ivjAuthorLabel1.setMaximumSize(new java.awt.Dimension(41, 23));
209             ivjAuthorLabel1.setMinimumSize(new java.awt.Dimension(41, 23));
210             // user code begin {1}
211             // user code end
212         } catch (java.lang.Throwable ivjExc) {
213             // user code begin {2}

```

```

214         // user code end
215         handleException(ivjExc);
216     }
217 }
218 return ivjAuthorLabel1;
}
220 /**
 * Return the CloseWindowItem property value.
221 * @return javax.swing.JMenuItem
 */
224 /* WARNING: THIS METHOD WILL BE REGENERATED. */
225 private javax.swing.JMenuItem getCloseWindowItem() {
226     if (ivjCloseWindowItem == null) {
227         try {
228             ivjCloseWindowItem = new javax.swing.JMenuItem();
229             ivjCloseWindowItem.setName("CloseWindowItem");
230             ivjCloseWindowItem.setText("CloseWindow");
231             // user code begin {1}
232             // user code end
233         } catch (java.lang.Throwable ivjExc) {
234             // user code begin {2}
235             // user code end
236             handleException(ivjExc);
237         }
238     }
239     return ivjCloseWindowItem;
240 }
241 /**
242 * Return the ComposeContribution property value.
243 * @return javax.swing.JDialog
244 */
245 /* WARNING: THIS METHOD WILL BE REGENERATED. */
246 private javax.swing.JDialog getComposeContribution() {
247     if (ivjComposeContribution == null) {
248         try {
249             ivjComposeContribution = new javax.swing.JDialog();
250             ivjComposeContribution.setName("ComposeContribution");
251             ivjComposeContribution.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
252             ivjComposeContribution.setBounds(40, 672, 564, 500);
253             ivjComposeContribution.setTitle("ComposeNewMessage");
254             getComposeContribution().setContentPane(getJDialogContentPane());
255             // user code begin {1}
256             // user code end
257         } catch (java.lang.Throwable ivjExc) {
258             // user code begin {2}
259             // user code end
260             handleException(ivjExc);
261         }
262     }
263     return ivjComposeContribution;
264 }
265 /**
266 * Return the ContentsItem property value.
267 * @return javax.swing.JMenuItem
268 */
269 /* WARNING: THIS METHOD WILL BE REGENERATED. */
270 private javax.swing.JMenuItem getContentsItem() {
271     if (ivjContentsItem == null) {
272         try {
273             ivjContentsItem = new javax.swing.JMenuItem();
274             ivjContentsItem.setName("ContentsItem");
275             ivjContentsItem.setText("Contents");
276             // user code begin {1}
277             // user code end
278         } catch (java.lang.Throwable ivjExc) {
279             // user code begin {2}

```

```

280         // user code end
281         handleException(ivjExc);
282     }
283     return ivjContentsItem;
284 }
285 /**
286 * Return the DateLabel property value.
287 * @return javax.swing.JLabel
288 */
289 /* WARNING: THIS METHOD WILL BE REGENERATED. */
290 private javax.swing.JLabel getDateLabel() {
291     if (ivjDateLabel == null) {
292         try {
293             ivjDateLabel = new javax.swing.JLabel();
294             ivjDateLabel.setName("DateLabel");
295             ivjDateLabel.setText("Date:");
296             ivjDateLabel.setMaximumSize(new java.awt.Dimension(29, 23));
297             ivjDateLabel.setMinimumSize(new java.awt.Dimension(29, 23));
298             // user code begin {1}
299             // user code end
300         } catch (java.lang.Throwable ivjExc) {
301             // user code begin {2}
302             // user code end
303             handleException(ivjExc);
304         }
305     }
306     return ivjDateLabel;
307 }
308 /**
309 * Return the DateLabel1 property value.
310 * @return javax.swing.JLabel
311 */
312 /* WARNING: THIS METHOD WILL BE REGENERATED. */
313 private javax.swing.JLabel getDateLabel1() {
314     if (ivjDateLabel1 == null) {
315         try {
316             ivjDateLabel1 = new javax.swing.JLabel();
317             ivjDateLabel1.setName("DateLabel1");
318             ivjDateLabel1.setPreferredSize(new java.awt.Dimension(29, 23));
319             ivjDateLabel1.setText("Date:");
320             ivjDateLabel1.setMaximumSize(new java.awt.Dimension(29, 23));
321             ivjDateLabel1.setMinimumSize(new java.awt.Dimension(29, 23));
322             // user code begin {1}
323             // user code end
324         } catch (java.lang.Throwable ivjExc) {
325             // user code begin {2}
326             // user code end
327             handleException(ivjExc);
328         }
329     }
330     return ivjDateLabel1;
331 }
332 /**
333 * Return the DefaultToolBarButton property value.
334 * @return javax.swing.JButton
335 */
336 /* WARNING: THIS METHOD WILL BE REGENERATED. */
337 private javax.swing.JButton getDefaultToolBarButton() {
338     if (ivjDefaultToolBarButton == null) {
339         try {
340             ivjDefaultToolBarButton = new javax.swing.JButton();
341             ivjDefaultToolBarButton.setName("DefaultToolBarButton");
342             ivjDefaultToolBarButton.setText("");
343             ivjDefaultToolBarButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
344             ivjDefaultToolBarButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);

```

```

346         ivjDefaultToolBarButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
            toolbarButtonGraphics/general/Add16.gif")));
347         ivjDefaultToolBarButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
348     // user code begin {1}
349     // user code end
350 } catch (java.lang.Throwable ivjExc) {
351     // user code begin {2}
352     // user code end
353     handleException(ivjExc);
354 }
355 }
356 return ivjDefaultToolBarButton;
357 }
358 /**
359 * Return the JButton2 property value.
360 * @return javax.swing.JButton
361 */
362 /* WARNING: THIS METHOD WILL BE REGENERATED. */
363 private javax.swing.JButton getDiscardButton() {
364     if (ivjDiscardButton == null) {
365         try {
366             ivjDiscardButton = new javax.swing.JButton();
367             ivjDiscardButton.setName("DiscardButton");
368             ivjDiscardButton.setText("Discard");
369             // user code begin {1}
370             // user code end
371         } catch (java.lang.Throwable ivjExc) {
372             // user code begin {2}
373             // user code end
374             handleException(ivjExc);
375         }
376     }
377     return ivjDiscardButton;
378 }
379 /**
380 * Return the DiscussionFrameJMenuBar property value.
381 * @return javax.swing.JMenuBar
382 */
383 /* WARNING: THIS METHOD WILL BE REGENERATED. */
384 private javax.swing.JMenuBar getDiscussionFrameJMenuBar() {
385     if (ivjDiscussionFrameJMenuBar == null) {
386         try {
387             ivjDiscussionFrameJMenuBar = new javax.swing.JMenuBar();
388             ivjDiscussionFrameJMenuBar.setName("DiscussionFrameJMenuBar");
389             ivjDiscussionFrameJMenuBar.add(getDiscussionMenu());
390             ivjDiscussionFrameJMenuBar.add(getHelpMenu());
391             // user code begin {1}
392             // user code end
393         } catch (java.lang.Throwable ivjExc) {
394             // user code begin {2}
395             // user code end
396             handleException(ivjExc);
397         }
398     }
399     return ivjDiscussionFrameJMenuBar;
400 }
401 /**
402 * Return the DiscussionMenu property value.
403 * @return javax.swing.JMenu
404 */
405 /* WARNING: THIS METHOD WILL BE REGENERATED. */
406 private javax.swing.JMenu getDiscussionMenu() {
407     if (ivjDiscussionMenu == null) {
408         try {
409             ivjDiscussionMenu = new javax.swing.JMenu();
410             ivjDiscussionMenu.setName("DiscussionMenu");

```

```

        ivjDiscussionMenu.setText("Discuss");
412    ivjDiscussionMenu.add(getNewThreadItem());
413    ivjDiscussionMenu.add(getNewMessageItem());
414    ivjDiscussionMenu.add(getJSeparator1());
415    ivjDiscussionMenu.add(getCloseWindowItem());
416    // user code begin {1}
417    // user code end
418 } catch (java.lang.Throwable ivjExc) {
419     // user code begin {2}
420     // user code end
421     handleException(ivjExc);
422 }
423 }
424 return ivjDiscussionMenu;
}
425 /**
 * Return the DiscussionTree property value.
426 * @return javax.swing.JTree
 */
427 /* WARNING: THIS METHOD WILL BE REGENERATED. */
428 private javax.swing.JTree getDiscussionTree() {
429     if (ivjDiscussionTree == null) {
430         try {
431             ivjDiscussionTree = new javax.swing.JTree();
432             ivjDiscussionTree.setName("DiscussionTree");
433             ivjDiscussionTree.setToolTipText("Discussion\u00e9tree");
434             ivjDiscussionTree.setPreferredSize(new java.awt.Dimension(78, 200));
435             ivjDiscussionTree.setMaximumSize(new java.awt.Dimension(100000, 100000));
436             // user code begin {1}
437             // user code end
438         } catch (java.lang.Throwable ivjExc) {
439             // user code begin {2}
440             // user code end
441             handleException(ivjExc);
442         }
443     }
444     return ivjDiscussionTree;
445 }
446 /**
447 * Return the HelpMenu property value.
448 * @return javax.swing.JMenu
449 */
450 /* WARNING: THIS METHOD WILL BE REGENERATED. */
451 private javax.swing.JMenu getHelpMenu() {
452     if (ivjHelpMenu == null) {
453         try {
454             ivjHelpMenu = new javax.swing.JMenu();
455             ivjHelpMenu.setName("HelpMenu");
456             ivjHelpMenu.setText("Help");
457             ivjHelpMenu.add(getContentsItem());
458             ivjHelpMenu.add(getAboutItem());
459             // user code begin {1}
460             // user code end
461         } catch (java.lang.Throwable ivjExc) {
462             // user code begin {2}
463             // user code end
464             handleException(ivjExc);
465         }
466     }
467     return ivjHelpMenu;
468 }
469 /**
470 * Return the JDialogContentPane property value.
471 * @return javax.swing.JPanel
472 */
473 /* 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(), getName());
484         getJDialogContentPane().add(get JPanel3(), getName());
        getJDialogContentPane().add(get JPanel4(), 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 */
516 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getJFrameContentPane() {
    if (ivjJFrameContentPane == null) {
        try {
518         ivjJFrameContentPane = new javax.swing.JPanel();
        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();
543     ivjJLabel1.setName("JLabel1");
544     ivjJLabel1.setText("Jlabel1");
545     ivjJLabel1.setForeground(java.awt.SystemColor.controlText);
546     ivjJLabel1.setMaximumSize(new java.awt.Dimension(45, 23));
547     ivjJLabel1.setMinimumSize(new java.awt.Dimension(45, 23));
548     // user code begin {1}
549     // user code end
550   } catch (java.lang.Throwable ivjExc) {
551     // user code begin {2}
552     // user code end
553     handleException(ivjExc);
554   }
555 }
556 return ivjJLabel1;
557 */
558 /**
559 * Return the JLabel2 property value.
560 * @return javax.swing.JLabel
561 */
562 /* WARNING: THIS METHOD WILL BE REGENERATED. */
563 private javax.swing.JLabel getJLabel2() {
564   if (ivjJLabel2 == null) {
565     try {
566       ivjJLabel2 = new javax.swing.JLabel();
567       ivjJLabel2.setName("JLabel2");
568       ivjJLabel2.setText("JLabel2");
569       ivjJLabel2.setMaximumSize(new java.awt.Dimension(45, 23));
570       ivjJLabel2.setForeground(java.awt.SystemColor.controlText);
571       ivjJLabel2.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
572       ivjJLabel2.setMinimumSize(new java.awt.Dimension(45, 23));
573       // user code begin {1}
574       // user code end
575     } catch (java.lang.Throwable ivjExc) {
576       // user code begin {2}
577       // user code end
578       handleException(ivjExc);
579     }
580   }
581   return ivjJLabel2;
582 }
583 /**
584 * Return the JPanel1 property value.
585 * @return javax.swing.JPanel
586 */
587 /* WARNING: THIS METHOD WILL BE REGENERATED. */
588 private javax.swing.JPanel getJPanel1() {
589   if (ivjJPanel1 == null) {
590     try {
591       ivjJPanel1 = new javax.swing.JPanel();
592       ivjJPanel1.setName("JPanel1");
593       ivjJPanel1.setLayout(new java.awt.BorderLayout());
594       getJPanel1().add(getDiscussionTree(), "Center");
595       // user code begin {1}
596       // user code end
597     } catch (java.lang.Throwable ivjExc) {
598       // user code begin {2}
599       // user code end
600       handleException(ivjExc);
601     }
602   }
603   return ivjJPanel1;
604 }
605 /**
606 * Return the JPanel2 property value.
607 * @return javax.swing.JPanel

```

```

608  /*
609   * WARNING: THIS METHOD WILL BE REGENERATED. */
610  private javax.swing.JPanel getJPanel2() {
611      if (ivj JPanel2 == null) {
612          try {
613              ivj JPanel2 = new javax.swing.JPanel();
614              ivj JPanel2.setName("JPanel2");
615              ivj JPanel2.setLayout(getJPanel2BoxLayout());
616              getJPanel2().add(getPanelHoldingHeader(), getPanelHoldingHeader().getName());
617              getJPanel2().add(getMessageDisplayArea(), getMessageDisplayArea().getName());
618              // user code begin {1}
619              // user code end
620          } catch (java.lang.Throwable ivjExc) {
621              // user code begin {2}
622              // user code end
623              handleException(ivjExc);
624          }
625      }
626      return ivj JPanel2;
627  /**
628   * Return the JPanel2BoxLayout property value.
629   * @return javax.swing.BoxLayout
630   */
631  /* WARNING: THIS METHOD WILL BE REGENERATED. */
632  private javax.swing.BoxLayout getJPanel2BoxLayout() {
633      javax.swing.BoxLayout ivj JPanel2BoxLayout = null;
634      try {
635          /* Create part */
636          ivj JPanel2BoxLayout = new javax.swing.BoxLayout(getJPanel2(), javax.swing.BoxLayout.Y_AXIS);
637      } catch (java.lang.Throwable ivjExc) {
638          handleException(ivjExc);
639      };
640      return ivj JPanel2BoxLayout;
641  }
642  /**
643   * Return the JPanel3 property value.
644   * @return javax.swing.JPanel
645   */
646  /* WARNING: THIS METHOD WILL BE REGENERATED. */
647  private javax.swing.JPanel getJPanel3() {
648      if (ivj JPanel3 == null) {
649          try {
650              ivj JPanel3 = new javax.swing.JPanel();
651              ivj JPanel3.setName("JPanel3");
652              ivj JPanel3.setLayout(getJPanel3FlowLayout());
653              getJPanel3().add(getSubmitButton(), getSubmitButton().getName());
654              getJPanel3().add(getDiscardButton(), getDiscardButton().getName());
655              // user code begin {1}
656              // user code end
657          } catch (java.lang.Throwable ivjExc) {
658              // user code begin {2}
659              // user code end
660              handleException(ivjExc);
661          }
662      }
663      return ivj JPanel3;
664  }
665  /**
666   * Return the JPanel3FlowLayout property value.
667   * @return java.awt.FlowLayout
668   */
669  /* WARNING: THIS METHOD WILL BE REGENERATED. */
670  private java.awt.FlowLayout getJPanel3FlowLayout() {
671      java.awt.FlowLayout ivj JPanel3FlowLayout = null;
672      try {

```

```

674     /* Create part */
675     ivj JPanel3FlowLayout = new java.awt.FlowLayout();
676     ivj JPanel3FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
677 } catch (java.lang.Throwable ivjExc) {
678     handleException(ivjExc);
679 }
680 return ivj JPanel3FlowLayout;
}
682 /**
683 * Return the JPanel4 property value.
684 * @return javax.swing.JPanel
685 */
686 /* WARNING: THIS METHOD WILL BE REGENERATED. */
687 private javax.swing.JPanel get JPanel4() {
688     if (ivj JPanel4 == null) {
689         try {
690             ivj JPanel4 = new javax.swing.JPanel();
691             ivj JPanel4.setName(" JPanel4");
692             ivj JPanel4.setLayout(null);
693             // user code begin {1}
694             // user code end
695             } catch (java.lang.Throwable ivjExc) {
696                 // user code begin {2}
697                 // user code end
698                 handleException(ivjExc);
699             }
700         }
701         return ivj JPanel4;
702     }
703 /**
704 * Return the JSeparator1 property value.
705 * @return javax.swing.JSeparator
706 */
707 /* WARNING: THIS METHOD WILL BE REGENERATED. */
708 private javax.swing.JSeparator get JSeparator1() {
709     if (ivj JSeparator1 == null) {
710         try {
711             ivj JSeparator1 = new javax.swing.JSeparator();
712             ivj JSeparator1.setName(" JSeparator1");
713             // user code begin {1}
714             // user code end
715             } catch (java.lang.Throwable ivjExc) {
716                 // user code begin {2}
717                 // user code end
718                 handleException(ivjExc);
719             }
720         }
721         return ivj JSeparator1;
722     }
723 /**
724 * Return the JSplitPane1 property value.
725 * @return javax.swing.JSplitPane
726 */
727 /* WARNING: THIS METHOD WILL BE REGENERATED. */
728 private javax.swing.JSplitPane get JSplitPane1() {
729     if (ivj JSplitPane1 == null) {
730         try {
731             ivj JSplitPane1 = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
732             ivj JSplitPane1.setName(" JSplitPane1");
733             ivj JSplitPane1.setDividerLocation(200);
734             get JSplitPane1().add(get JPanel1(), "top");
735             get JSplitPane1().add(get JPanel2(), "bottom");
736             // user code begin {1}
737             // user code end
738             } catch (java.lang.Throwable ivjExc) {
739                 // user code begin {2}

```

```

740         // user code end
741         handleException(ivjExc);
742     }
743 }
744 return ivjJSplitPane1;
745 }
746 /**
747 * Return the JTextField1 property value.
748 * @return javax.swing.JTextField
749 */
750 /* WARNING: THIS METHOD WILL BE REGENERATED. */
751 private javax.swing.JTextField getJTextField1() {
752     if (ivjJTextField1 == null) {
753         try {
754             ivjJTextField1 = new javax.swing.JTextField();
755             ivjJTextField1.setName("JTextField1");
756             ivjJTextField1.setPreferredSize(new java.awt.Dimension(4, 23));
757             ivjJTextField1.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null,
758                                         null));
759             ivjJTextField1.setMaximumSize(new java.awt.Dimension(2147483647, 23));
760             ivjJTextField1.setMinimumSize(new java.awt.Dimension(4, 23));
761             ivjJTextField1.setEditable(false);
762             // user code begin {1}
763             // user code end
764         } catch (java.lang.Throwable ivjExc) {
765             // user code begin {2}
766             // user code end
767             handleException(ivjExc);
768         }
769     }
770     return ivjJTextField1;
771 }
772 /**
773 * Return the JTextField11 property value.
774 * @return javax.swing.JTextField
775 */
776 /* WARNING: THIS METHOD WILL BE REGENERATED. */
777 private javax.swing.JTextField getJTextField11() {
778     if (ivjJTextField11 == null) {
779         try {
780             ivjJTextField11 = new javax.swing.JTextField();
781             ivjJTextField11.setName("JTextField11");
782             ivjJTextField11.setBorder(new javax.swing.plaf.BorderUIResource.CompoundBorderUIResource(null,
783                                         null));
784             ivjJTextField11.setMaximumSize(new java.awt.Dimension(2147483647, 23));
785             ivjJTextField11.setPreferredSize(new java.awt.Dimension(4, 23));
786             ivjJTextField11.setMinimumSize(new java.awt.Dimension(4, 23));
787             ivjJTextField11.setEditable(false);
788             // user code begin {1}
789             // user code end
790         } catch (java.lang.Throwable ivjExc) {
791             // user code begin {2}
792             // user code end
793             handleException(ivjExc);
794         }
795     }
796     return ivjJTextField11;
797 }
798 /**
799 * Return the JTextField12 property value.
800 * @return javax.swing.JTextField
801 */
802 /* WARNING: THIS METHOD WILL BE REGENERATED. */
803 private javax.swing.JTextField getJTextField12() {
804     if (ivjJTextField12 == null) {
805         try {

```

```

804     ivjJTextField12 = new javax.swing.JTextField();
805     ivjJTextField12.setName("JTextField12");
806     ivjJTextField12.setPreferredSize(new java.awt.Dimension(4, 23));
807     ivjJTextField12.setMaximumSize(new java.awt.Dimension(2147483647, 23));
808     ivjJTextField12.setMinimumSize(new java.awt.Dimension(4, 23));
809     // user code begin {1}
810     // user code end
811     } catch (java.lang.Throwable ivjExc) {
812         // user code begin {2}
813         // user code end
814         handleException(ivjExc);
815     }
816 }
817 return ivjJTextField12;
818 }
819 /**
820 * Return the JToolBar1 property value.
821 * @return javax.swing.JToolBar
822 */
823 /* WARNING: THIS METHOD WILL BE REGENERATED. */
824 private javax.swing.JToolBar getJToolBar1() {
825     if (ivjJToolBar1 == null) {
826         try {
827             ivjJToolBar1 = new javax.swing.JToolBar();
828             ivjJToolBar1.setName("JToolBar1");
829             getJToolBar1().add(getJToolBarButton1(), getJToolBarButton1().getName());
830             ivjJToolBar1.add(getDefaultToolBarButton());
831             ivjJToolBar1.addSeparator();
832             getJToolBar1().add(getJToolBarButton2(), getJToolBarButton2().getName());
833             getJToolBar1().add(getJToolBarButton3(), getJToolBarButton3().getName());
834             // user code begin {1}
835             // user code end
836         } catch (java.lang.Throwable ivjExc) {
837             // user code begin {2}
838             // user code end
839             handleException(ivjExc);
840         }
841     }
842     return ivjJToolBar1;
843 }
844 /**
845 * Return the JToolBarButton1 property value.
846 * @return javax.swing.JButton
847 */
848 /* WARNING: THIS METHOD WILL BE REGENERATED. */
849 private javax.swing.JButton getJToolBarButton1() {
850     if (ivjJToolBarButton1 == null) {
851         try {
852             ivjJToolBarButton1 = new javax.swing.JButton();
853             ivjJToolBarButton1.setName("JToolBarButton1");
854             ivjJToolBarButton1.setText("");
855             ivjJToolBarButton1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
856             ivjJToolBarButton1.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
857             ivjJToolBarButton1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
858                 toolbarButtonGraphics/general/New16.gif")));
859             ivjJToolBarButton1.setMargin(new java.awt.Insets(0, 0, 0, 0));
860             ivjJToolBarButton1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
861             // user code begin {1}
862             // user code end
863         } catch (java.lang.Throwable ivjExc) {
864             // user code begin {2}
865             // user code end
866             handleException(ivjExc);
867         }
868     }
869     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) {
877         try {
878             ivjJToolBarButton2 = new javax.swing.JButton();
879             ivjJToolBarButton2.setName("JToolBarButton2");
880             ivjJToolBarButton2.setText("");
881             ivjJToolBarButton2.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
882             ivjJToolBarButton2.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
883             ivjJToolBarButton2.setIcon(new javax.swing.ImageIcon(getClass().getResource(
884                 "toolbarButtonGraphics/general/Help16.gif")));
885             ivjJToolBarButton2.setMargin(new java.awt.Insets(0, 0, 0, 0));
886             // user code begin {1}
887             // user code end
888         } catch (java.lang.Throwable ivjExc) {
889             // user code begin {2}
890             // user code end
891             handleException(ivjExc);
892         }
893     }
894     return ivjJToolBarButton2;
}
895 /**
896 * Return the JToolBarButton3 property value.
897 * @return javax.swing.JButton
 */
898 /* WARNING: THIS METHOD WILL BE REGENERATED. */
900 private javax.swing.JButton getJToolBarButton3() {
901     if (ivjJToolBarButton3 == null) {
902         try {
903             ivjJToolBarButton3 = new javax.swing.JButton();
904             ivjJToolBarButton3.setName("JToolBarButton3");
905             ivjJToolBarButton3.setText("");
906             ivjJToolBarButton3.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
907             ivjJToolBarButton3.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
908             ivjJToolBarButton3.setIcon(new javax.swing.ImageIcon(getClass().getResource(
909                 "toolbarButtonGraphics/general/Stop16.gif")));
910             ivjJToolBarButton3.setMargin(new java.awt.Insets(0, 0, 0, 0));
911             // user code begin {1}
912             // user code end
913         } catch (java.lang.Throwable ivjExc) {
914             // user code begin {2}
915             // user code end
916             handleException(ivjExc);
917         }
918     }
919     return ivjJToolBarButton3;
}
920 /**
 * Return the MessageDisplayArea property value.
921 * @return javax.swing.JTextPane
 */
923 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JTextPane getMessageDisplayArea() {
924     if (ivjMessageDisplayArea == null) {
925         try {
926             ivjMessageDisplayArea = new javax.swing.JTextPane();
927             ivjMessageDisplayArea.setName("MessageDisplayArea");
928             ivjMessageDisplayArea.setToolTipText("Message display area");
929             ivjMessageDisplayArea.setEnabled(false);
930             // user code begin {1}
931

```

```

934         ivjMessageDisplayArea.setBorder(BorderFactory.createEtchedBorder());
935         // user code end
936     } catch (java.lang.Throwable ivjExc) {
937         // user code begin {2}
938         // user code end
939         handleException(ivjExc);
940     }
941     return ivjMessageDisplayArea;
942 }
943 /**
944 * Return the MessageDisplayArea1 property value.
945 * @return javax.swing.JTextPane
946 */
947 /* WARNING: THIS METHOD WILL BE REGENERATED. */
948 private javax.swing.JTextPane getMessageDisplayAreaOnComposeBox() {
949     if (ivjMessageDisplayAreaOnComposeBox == null) {
950         try {
951             ivjMessageDisplayAreaOnComposeBox = new javax.swing.JTextPane();
952             ivjMessageDisplayAreaOnComposeBox.setName("MessageDisplayAreaOnComposeBox");
953             ivjMessageDisplayAreaOnComposeBox.setToolTipText("MessageCompositionArea");
954             ivjMessageDisplayAreaOnComposeBox.setEnabled(true);
955             // user code begin {1}
956             ivjMessageDisplayAreaOnComposeBox.setBorder(BorderFactory.createEtchedBorder());
957             // user code end
958         } catch (java.lang.Throwable ivjExc) {
959             // user code begin {2}
960             // user code end
961             handleException(ivjExc);
962         }
963     }
964     return ivjMessageDisplayAreaOnComposeBox;
965 }
966 /**
967 * Return the MessagePanel property value.
968 * @return javax.swing.JPanel
969 */
970 /* WARNING: THIS METHOD WILL BE REGENERATED. */
971 private javax.swing.JPanel getMessagePanel() {
972     if (ivjMessagePanel == null) {
973         try {
974             ivjMessagePanel = new javax.swing.JPanel();
975             ivjMessagePanel.setName("MessagePanel");
976             ivjMessagePanel.setLayout(getMessagePanelBoxLayout());
977             ivjMessagePanel.add(getPanelHoldingHeaderOnComposeBox());
978             getMessagePanel().add(getMessageDisplayAreaOnComposeBox(), getMessageDisplayAreaOnComposeBox().
979                 getName());
980             // user code begin {1}
981             // user code end
982         } catch (java.lang.Throwable ivjExc) {
983             // user code begin {2}
984             // user code end
985             handleException(ivjExc);
986         }
987     }
988     return ivjMessagePanel;
989 }
990 /**
991 * Return the MessagePanelBoxLayout property value.
992 * @return javax.swing.BoxLayout
993 */
994 /* WARNING: THIS METHOD WILL BE REGENERATED. */
995 private javax.swing.BoxLayout getMessagePanelBoxLayout() {
996     javax.swing.BoxLayout ivjMessagePanelBoxLayout = null;
997     try {
998         /* Create part */

```

```

998     ivjMessagePanelBoxLayout = new javax.swing.BoxLayout(getMessagePanel(), javax.swing.BoxLayout.
    Y_AXIS);
1000    } catch (java.lang.Throwable ivjExc) {
1001        handleException(ivjExc);
1002    }
1003    return ivjMessagePanelBoxLayout;
1004 }
1005 /**
1006 * Return the NewMessageItem property value.
1007 * @return javax.swing.JMenuItem
1008 */
1009 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1010 private javax.swing.JMenuItem getNewMessageItem() {
1011    if (ivjNewMessageItem == null) {
1012        try {
1013            ivjNewMessageItem = new javax.swing.JMenuItem();
1014            ivjNewMessageItem.setName("NewMessageItem");
1015            ivjNewMessageItem.setText("NewMessage...");
1016            // user code begin {1}
1017            // user code end
1018        } catch (java.lang.Throwable ivjExc) {
1019            // user code begin {2}
1020            // user code end
1021            handleException(ivjExc);
1022        }
1023    }
1024    return ivjNewMessageItem;
1025 }
1026 /**
1027 * Return the NewThreadItem property value.
1028 * @return javax.swing.JMenuItem
1029 */
1030 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1031 private javax.swing.JMenuItem getNewThreadItem() {
1032    if (ivjNewThreadItem == null) {
1033        try {
1034            ivjNewThreadItem = new javax.swing.JMenuItem();
1035            ivjNewThreadItem.setName("NewThreadItem");
1036            ivjNewThreadItem.setText("NewThread");
1037            // user code begin {1}
1038            // user code end
1039        } catch (java.lang.Throwable ivjExc) {
1040            // user code begin {2}
1041            // user code end
1042            handleException(ivjExc);
1043        }
1044    }
1045    return ivjNewThreadItem;
1046 }
1047 /**
1048 * Return the PanelHoldingHeader property value.
1049 * @return javax.swing.JPanel
1050 */
1051 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1052 private javax.swing.JPanel getPanelHoldingHeader() {
1053    if (ivjPanelHoldingHeader == null) {
1054        try {
1055            ivjPanelHoldingHeader = new javax.swing.JPanel();
1056            ivjPanelHoldingHeader.setName("PanelHoldingHeader");
1057            ivjPanelHoldingHeader.setLayout(new javax.swing.BoxLayout(getPanelHoldingHeader(), javax.swing.
    BoxLayout.X_AXIS));
1058            ivjPanelHoldingHeader.setMaximumSize(new java.awt.Dimension(496, 120));
1059            ivjPanelHoldingHeader.setPreferredSize(new java.awt.Dimension(496, 70));
1060            ivjPanelHoldingHeader.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
1061            ivjPanelHoldingHeader.setMinimumSize(new java.awt.Dimension(496, 120));
1062        }
1063    }
1064    return ivjPanelHoldingHeader;
1065 }

```

```

        getPanelHoldingHeader().add(getPanelHoldingHeadingLabels(), getPanelHoldingHeadingLabels().
            getName());
    1062     getPanelHoldingHeader().add(getPanelHoldingHeaderFields(), getPanelHoldingHeaderFields().getName
        ());
    1064     // user code begin {1}
    1065     // user code end
    1066 } catch (java.lang.Throwable ivjExc) {
    1067     // user code begin {2}
    1068     // user code end
    1069     handleException(ivjExc);
    1070 }
    1071 return ivjPanelHoldingHeader;
1072 }
1073 /**
 * Return the PanelHoldingHeaderFields property value.
 * @return javax.swing.JPanel
1076 */
1077 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1078 private javax.swing.JPanel getPanelHoldingHeaderFields() {
    1079     if (ivjPanelHoldingHeaderFields == null) {
    1080         try {
    1081             ivjPanelHoldingHeaderFields = new javax.swing.JPanel();
    1082             ivjPanelHoldingHeaderFields.setName("PanelHoldingHeaderFields");
    1083             ivjPanelHoldingHeaderFields.setLayout(getPanelHoldingHeaderFieldsBoxLayout());
    1084             ivjPanelHoldingHeaderFields.setMaximumSize(new java.awt.Dimension(450, 69));
    1085             ivjPanelHoldingHeaderFields.setMinimumSize(new java.awt.Dimension(450, 69));
    1086             getPanelHoldingHeaderFields().add(getAuthor(), getAuthor().getName());
    1087             getPanelHoldingHeaderFields().add(getJLabel2(), getJLabel2().getName());
    1088             getPanelHoldingHeaderFields().add(getJLabel1(), getJLabel1().getName());
    1089             // user code begin {1}
    1090             ivjPanelHoldingHeaderFields.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
    1091                 ivjPanelHoldingHeaderFields.getBackground()));
    1092             // user code end
    1093         } catch (java.lang.Throwable ivjExc) {
    1094             // user code begin {2}
    1095             // user code end
    1096             handleException(ivjExc);
    1097         }
    1098     return ivjPanelHoldingHeaderFields;
    1099 }
1100 /**
 * Return the PanelHoldingHeaderFieldsBoxLayout property value.
 * @return javax.swing.BoxLayout
 */
1101 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1102 private javax.swing.BoxLayout getPanelHoldingHeaderFieldsBoxLayout() {
    1103     javax.swing.BoxLayout ivjPanelHoldingHeaderFieldsBoxLayout = null;
    1104     try {
    1105         /* Create part */
    1106         ivjPanelHoldingHeaderFieldsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeaderFields(),
    1107             javax.swing.BoxLayout.Y_AXIS);
    1108     } catch (java.lang.Throwable ivjExc) {
    1109         handleException(ivjExc);
    1110     };
    1111     return ivjPanelHoldingHeaderFieldsBoxLayout;
    1112 }
1113 /**
 * Return the PanelHoldingHeaderFields1 property value.
 * @return javax.swing.JPanel
 */
1115 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1116 private javax.swing.JPanel getPanelHoldingHeaderFieldsOnComposeBox() {
    1117     if (ivjPanelHoldingHeaderFieldsOnComposeBox == null) {
    1118         try {

```

```

1124     ivjPanelHoldingHeaderFieldsOnComposeBox = new javax.swing.JPanel();
1125     ivjPanelHoldingHeaderFieldsOnComposeBox.setName("PanelHoldingHeaderFieldsOnComposeBox");
1126     ivjPanelHoldingHeaderFieldsOnComposeBox.setPreferredSize(new java.awt.Dimension(470000, 75));
1127     ivjPanelHoldingHeaderFieldsOnComposeBox.setLayout(
1128         getPanelHoldingHeaderFieldsOnComposeBoxBoxLayout());
1129     ivjPanelHoldingHeaderFieldsOnComposeBox.setMaximumSize(new java.awt.Dimension(45000, 75));
1130     ivjPanelHoldingHeaderFieldsOnComposeBox.setMinimumSize(new java.awt.Dimension(450, 75));
1131     ivjPanelHoldingHeaderFieldsOnComposeBox.add(getJTextField1());
1132     getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField11(), getJTextField11().getName());
1133     getPanelHoldingHeaderFieldsOnComposeBox().add(getJTextField12(), getJTextField12().getName());
1134     // user code begin {1}
1135     ivjPanelHoldingHeaderFieldsOnComposeBox.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
1136         ivjPanelHoldingHeaderFieldsOnComposeBox.getBackground()));
1137     // user code end
1138 } catch (java.lang.Throwable ivjExc) {
1139     // user code begin {2}
1140     // user code end
1141     handleException(ivjExc);
1142 }
1143 return ivjPanelHoldingHeaderFieldsOnComposeBox;
}
1144 /**
 * Return the PanelHoldingHeaderFieldsOnComposeBoxBoxLayout property value.
1145 * @return javax.swing.BoxLayout
 */
1146 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.BoxLayout getPanelHoldingHeaderFieldsOnComposeBoxBoxLayout() {
1147     javax.swing.BoxLayout ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = null;
1148     try {
1149         /* Create part */
1150         ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
1151             getPanelHoldingHeaderFieldsOnComposeBox(), javax.swing.BoxLayout.Y_AXIS);
1152     } catch (java.lang.Throwable ivjExc) {
1153         handleException(ivjExc);
1154     };
1155     return ivjPanelHoldingHeaderFieldsOnComposeBoxBoxLayout;
1156 }
1157 /**
 * Return the PanelHoldingHeader1 property value.
 * @return javax.swing.JPanel
 */
1158 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1159 private javax.swing.JPanel getPanelHoldingHeaderOnComposeBox() {
1160     if (ivjPanelHoldingHeaderOnComposeBox == null) {
1161         try {
1162             ivjPanelHoldingHeaderOnComposeBox = new javax.swing.JPanel();
1163             ivjPanelHoldingHeaderOnComposeBox.setName("PanelHoldingHeaderOnComposeBox");
1164             ivjPanelHoldingHeaderOnComposeBox.setLayout(new javax.swing.BoxLayout(
1165                 getPanelHoldingHeaderOnComposeBox(), javax.swing.BoxLayout.X_AXIS));
1166             ivjPanelHoldingHeaderOnComposeBox.setMaximumSize(new java.awt.Dimension(496000, 120));
1167             ivjPanelHoldingHeaderOnComposeBox.setPreferredSize(new java.awt.Dimension(496000, 90));
1168             ivjPanelHoldingHeaderOnComposeBox.setAlignmentX(java.awt.Component.LEFT_ALIGNMENT);
1169             ivjPanelHoldingHeaderOnComposeBox.setMinimumSize(new java.awt.Dimension(496, 120));
1170             getPanelHoldingHeaderOnComposeBox().add(getPanelHoldingHeadingLabelsOnComposeBox(),
1171                 getPanelHoldingHeadingLabelsOnComposeBox().getName());
1172             getPanelHoldingHeaderOnComposeBox().add(getPanelHoldingHeaderFieldsOnComposeBox(),
1173                 getPanelHoldingHeaderFieldsOnComposeBox().getName());
1174             // user code begin {1}
1175             // user code end
1176         } catch (java.lang.Throwable ivjExc) {
1177             // user code begin {2}
1178             // user code end
1179             handleException(ivjExc);
1180         }
1181     }

```

```

        }
1184     return ivjPanelHoldingHeaderOnComposeBox ;
}
1186 /**
 * Return the PanelHoldingHeadingLabels property value.
1188 * @return javax.swing.JPanel
 */
1190 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeadingLabels() {
1192     if (ivjPanelHoldingHeadingLabels == null) {
        try {
1194         ivjPanelHoldingHeadingLabels = new javax.swing.JPanel();
        ivjPanelHoldingHeadingLabels.setName("PanelHoldingHeadingLabels");
1196         ivjPanelHoldingHeadingLabels.setLayout(getPanelHoldingHeadingLabelsBoxLayout());
        getPanelHoldingHeadingLabels().add(getAuthorLabel(), getAuthorLabel().getName());
1198         getPanelHoldingHeadingLabels().add(getDateLabel(), getDateLabel().getName());
        getPanelHoldingHeadingLabels().add(getSubjectLabel(), getSubjectLabel().getName());
1200         // user code begin {1}
        ivjPanelHoldingHeadingLabels.setBorder(BorderFactory.createMatteBorder(6,6,6,6,
            ivjPanelHoldingHeadingLabels.getBackground()));
1202         // user code end
    } catch (java.lang.Throwable ivjExc) {
1204         // user code begin {2}
        // user code end
1206         handleException(ivjExc);
    }
1208     }
     return ivjPanelHoldingHeadingLabels;
1210 }
*/
1212 /**
 * Return the PanelHoldingHeadingLabelsBoxLayout property value.
 * @return javax.swing.BoxLayout
 */
1214 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
1216 private javax.swing.BoxLayout getPanelHoldingHeadingLabelsBoxLayout() {
    javax.swing.BoxLayout ivjPanelHoldingHeadingLabelsBoxLayout = null;
1218     try {
        /* Create part */
1220     ivjPanelHoldingHeadingLabelsBoxLayout = new javax.swing.BoxLayout(getPanelHoldingHeadingLabels(),
        javax.swingBoxLayout.Y_AXIS);
    } catch (java.lang.Throwable ivjExc) {
1222         handleException(ivjExc);
    };
     return ivjPanelHoldingHeadingLabelsBoxLayout;
}
1226 /**
 * Return the PanelHoldingHeadingLabels1 property value.
1228 * @return javax.swing.JPanel
 */
1230 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JPanel getPanelHoldingHeadingLabelsOnComposeBox() {
1232     if (ivjPanelHoldingHeadingLabelsOnComposeBox == null) {
        try {
1234         ivjPanelHoldingHeadingLabelsOnComposeBox = new javax.swing.JPanel();
        ivjPanelHoldingHeadingLabelsOnComposeBox.setName("PanelHoldingHeadingLabelsOnComposeBox");
1236         ivjPanelHoldingHeadingLabelsOnComposeBox.setPreferredSize(new java.awt.Dimension(65, 70));
        ivjPanelHoldingHeadingLabelsOnComposeBox.setLayout(
            getPanelHoldingHeadingLabelsOnComposeBoxBoxLayout());
1238         ivjPanelHoldingHeadingLabelsOnComposeBox.setMaximumSize(new java.awt.Dimension(65, 75));
        ivjPanelHoldingHeadingLabelsOnComposeBox.setMinimumSize(new java.awt.Dimension(65, 75));
1240         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,
1245             ivjPanelHoldingHeadingLabelsOnComposeBox.getBackground()));
1246             // user code end
1247         } catch (java.lang.Throwable ivjExc) {
1248             // user code begin {2}
1249             // user code end
1250             handleException(ivjExc);
1251         }
1252     return ivjPanelHoldingHeadingLabelsOnComposeBox;
1253 }
1254 /**
1255 * Return the PanelHoldingHeadingLabelsOnComposeBoxBoxLayout property value.
1256 * @return javax.swing.BoxLayout
1257 */
1258 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1259 private javax.swing.BoxLayout getPanelHoldingHeadingLabelsOnComposeBoxBoxLayout() {
1260     javax.swing.BoxLayout ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = null;
1261     try {
1262         /* Create part */
1263         ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout = new javax.swing.BoxLayout(
1264             getPanelHoldingHeadingLabelsOnComposeBox(), javax.swing.BoxLayout.Y_AXIS);
1265     } catch (java.lang.Throwable ivjExc) {
1266         handleException(ivjExc);
1267     };
1268     return ivjPanelHoldingHeadingLabelsOnComposeBoxBoxLayout;
1269 }
1270 /**
1271 * Return the SubjectLabel property value.
1272 * @return javax.swing.JLabel
1273 */
1274 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1275 private javax.swing.JLabel getSubjectLabel() {
1276     if (ivjSubjectLabel == null) {
1277         try {
1278             ivjSubjectLabel = new javax.swing.JLabel();
1279             ivjSubjectLabel.setName("SubjectLabel");
1280             ivjSubjectLabel.setText("Subject:");
1281             ivjSubjectLabel.setMaximumSize(new java.awt.Dimension(46, 23));
1282             ivjSubjectLabel.setMinimumSize(new java.awt.Dimension(46, 23));
1283             // user code begin {1}
1284             // user code end
1285         } catch (java.lang.Throwable ivjExc) {
1286             // user code begin {2}
1287             // user code end
1288             handleException(ivjExc);
1289         }
1290     }
1291     return ivjSubjectLabel;
1292 }
1293 /**
1294 * Return the SubjectLabel1 property value.
1295 * @return javax.swing.JLabel
1296 */
1297 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1298 private javax.swing.JLabel getSubjectLabel1() {
1299     if (ivjSubjectLabel1 == null) {
1300         try {
1301             ivjSubjectLabel1 = new javax.swing.JLabel();
1302             ivjSubjectLabel1.setName("SubjectLabel1");
1303             ivjSubjectLabel1.setPreferredSize(new java.awt.Dimension(46, 23));
1304             ivjSubjectLabel1.setText("Subject:");
1305             ivjSubjectLabel1.setMaximumSize(new java.awt.Dimension(46, 23));
1306             ivjSubjectLabel1.setMinimumSize(new java.awt.Dimension(46, 23));
1307             // user code begin {1}
1308             // user code end

```

```

1308     } catch (java.lang.Throwable ivjExc) {
1309         // user code begin {2}
1310         // user code end
1311         handleException(ivjExc);
1312     }
1313 }
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) {
1323         try {
1324             ivjSubmitButton = new javax.swing.JButton();
1325             ivjSubmitButton.setName("SubmitButton");
1326             ivjSubmitButton.setText("Submit");
1327             // user code begin {1}
1328             // user code end
1329         } catch (java.lang.Throwable ivjExc) {
1330             // user code begin {2}
1331             // user code end
1332             handleException(ivjExc);
1333         }
1334     }
1335     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 -----");
1345     // exception.printStackTrace(System.out);
1346 }
1348 /**
 * Initializes connections
 * @exception java.lang.Exception The exception description.
 */
1350 */
1352 /* WARNING: THIS METHOD WILL BE REGENERATED. */
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 */
1362 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
    try {
        // user code begin {1}
        // user code end
1366        setName("DiscussionFrame");
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1368        setTitle("Discussion\u00d7xxxx.yyy");
        setSize(600, 600);
1370        setJMenuBar(getDiscussionFrameJMenuBar());
        setContentPane(getJFrameContentPane());
        initConnections();
    } catch (java.lang.Throwable ivjExc) {

```

```

1374         handleException(ivjExc);
1375     }
1376     // user code begin {2}
1377     // user code end
1378 }
1379 /**
1380 * main entrypoint - starts the part when it is run as an application
1381 * @param args java.lang.String[]
1382 */
1383 public static void main(java.lang.String[] args) {
1384     try {
1385         DiscussionFrame aDiscussionFrame;
1386         aDiscussionFrame = new DiscussionFrame();
1387         aDiscussionFrame.addWindowListener(new java.awt.event.WindowAdapter() {
1388             public void windowClosing(java.awt.event.WindowEvent e) {
1389                 System.exit(0);
1390             }
1391         });
1392         aDiscussionFrame.setVisible(true);
1393     } catch (Throwable exception) {
1394         System.err.println("Exception occurred in main() of javax.swing.JFrame");
1395         exception.printStackTrace(System.out);
1396     }
1397 }
1398 }
```

Listing C.11: GridLayout.java

```

package clientapp;
2

4 import java.awt.Dimension;
5 import java.lang.Exception;
6
7 /**
8 * Default layout class for the client panorama. Manages a rectangular grid ordering that ought to be
9     easily customizable.
10    * Creation date: (22-06-00 11:42:53)
11    * @author:
12    */
12 public class GridLayout extends PanoramaLayout {
13     private int width;
14     private int height;
15
16     /**
17      * GridLayout constructor comment.
18     */
18 public GridLayout()
19 {
20     super();
21     setHeight( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
22         getGRID_LAYOUT_DEFAULT_NUMBER_OF_COLUMNS() ) ) );
22     setWidth( Integer.parseInt( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
23         getGRID_LAYOUT_DEFAULT_NUMBER_OF_ROWS() ) ) );
24 }
24 /**
25     * Convert the vertical and horizontal indices into a x,y coordinate pair designating a position in the
26     * panorama.
27     * Creation date: (22-06-00 12:02:00)
28     * @return java.awt.Dimension
29     */
29 public java.awt.Dimension computePlacement() throws Exception
30 {
31     Dimension newPlacement = new Dimension( (int) ( horizontalIndex * (int)(preferredDocumentSize.
32         getWidth() + getXSpacing() ) ),
32             (int) ( verticalIndex * (int)(preferredDocumentSize.getHeight() +
33                 getYSpacing() ) ) );
33     if (horizontalIndex >= width)
34 }
```

```

34     {
35         if (verticalIndex >= height)
36         {
37             throw new Exception( ClientConstants.getOUT_OF_BOUNDS_EXCEPTION_TEXT() );
38         }
39         else
40         {
41             verticalIndex++;
42             horizontalIndex = 0;
43         }
44     }
45     else
46     {
47         horizontalIndex++;
48     }
49     return newPlacement;
50 }
51 /**
52 * Insert the method's description here.
53 * Creation date: (22-06-00 11:43:44)
54 * @return int
55 */
56 public int getHeight() {
57     return height;
58 }
59 /**
60 * Compute and return the dimensions of a rectangle containing all the document nodes being laid out.
61 * Creation date: (01-08-00 19:43:12)
62 * @return java.awt.Rectangle
63 */
64 public java.awt.Rectangle getOverviewRectangle()
65 {
66     java.awt.Rectangle newRectangle = new java.awt.Rectangle();
67     if (getVerticalIndex() == 0)
68     {
69         newRectangle.setRect( -( getXSpacing()/2 ), 0,
70             getHorizontalIndex() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ),
71             getVerticalIndex() * ( getPreferredDocumentSize().getHeight() + getYSpacing() ) );
72     }
73     else
74     {
75         newRectangle.setRect( -( getXSpacing()/2 ), 0,
76             getWidth() * ( getPreferredDocumentSize().getWidth() + getXSpacing() ),
77             getVerticalIndex() * ( getPreferredDocumentSize().getHeight() + getYSpacing() ) );
78     }
79     return newRectangle;
80 }
81 /**
82 * Insert the method's description here.
83 * Creation date: (22-06-00 11:43:29)
84 * @return int
85 */
86 public int getWidth() {
87     return width;
88 }
89 /**
90 * Insert the method's description here.
91 * Creation date: (23-06-00 22:17:36)
92 */
93 public void resetLayout()
94 {
95     horizontalIndex = 0;
96     verticalIndex = 0;
97 }
98 /**
99 * Insert the method's description here.

```

```

100 * Creation date: (22-06-00 11:43:44)
101 * @param newHeight int
102 */
103 public void setHeight(int newHeight) {
104     height = newHeight;
105 }
106 /**
107 * Insert the method's description here.
108 * Creation date: (22-06-00 11:43:29)
109 * @param newWidth int
110 */
111 public void setWidth(int newWidth) {
112     width = newWidth;
113 }
114 }
```

Listing C.12: GridPanoramaLayoutManager.java

```

package clientapp;
1
import java.util.ArrayList;
2
import java.awt.event.*;
3
import javax.swing.event.*;
4
5
/**
6 * This class implements a simple layout manager which arranges the panorama documents in matrix pattern
7 *
8 * Creation date: (28-08-00 23:56:47)
9 * @author:
10 */
11 public class GridPanoramaLayoutManager extends ClientPanoramaLayoutManager
{
12
13     private javax.swing.JPanel ivjButtonPanel = null;
14     private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
15     private javax.swing.JButton ivjCancelButton = null;
16     private javax.swing.JTextField ivjDocumentHeightField = null;
17     private javax.swing.JLabel ivjDocumentHeightLabel = null;
18     private javax.swing.JTextField ivjDocumentWidthField = null;
19     private javax.swing.JLabel ivjDocumentWidthLabel = null;
20     private javax.swing.JDialog ivjGridLayoutCustomizationBox = null;
21     private javax.swing.JTextField ivjHorizontalSpacingField = null;
22     private javax.swing.JLabel ivjHorizontalSpacingLabel = null;
23     private javax.swing.JPanel ivjJDialogContentPane = null;
24     private javax.swing.JPanel ivjMainPanel = null;
25     private javax.swing.JButton ivjOKButton = null;
26     private javax.swing.JTextField ivjVerticalSpacingField = null;
27     private javax.swing.JLabel ivjVerticalSpacingLabel = null;
28     private final static java.lang.String GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING =
29         "Grid_layout_document_horizontal_spacing";
30     private final static java.lang.String GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING =
31         "Grid_layout_document_vertical_spacing";
32     private final static int GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE = 200;
33     private final static int GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE = 200;
34
35     public class CancelHandler implements ActionListener
36     {
37         public void actionPerformed( ActionEvent ae )
38         {
39             setCommitted( false );
40             getGridLayoutCustomizationBox().dispose();
41         }
42     };
43
44     public class OKHandler implements ActionListener
45     {
46         public void actionPerformed( ActionEvent ae )
47         {
```

```

        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";
53 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;
55 private final static int GRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE = 600;
56 /**
 * GridPanoramaLayout constructor comment.
57 */
58 public GridPanoramaLayoutManager() {
59     super();
60     initialize();
61 }
62 /**
63 * Insert the method's description here.
64 * Creation date: (12-10-00 11:50:40)
65 * @return java.lang.Object
66 */
67 public Object clone() {
68     return null;
69 }
70 /**
71 * Computes a list of Point objects representing a grid structure on which the nodes can subsequently be
72 * distributed.
73 */
74 public java.util.ArrayList computeCoordinates(edu.umd.cs.jazz.ZNode[] nodes)
75 {
76     int xCoordinate = 0;
77     int yCoordinate = 0;
78
79     int xIncrement = 0;
80     int yIncrement = 0;
81     // let document with the largest bounding box determine the spacing between all the documents
82     for ( int i = 0; i < nodes.length; i++ )
83     {
84         xIncrement = (int)Math.round( nodes[i].getBounds().getWidth() ) > xIncrement ? (int)Math.round(
85             nodes[i].getBounds().getWidth() ): xIncrement;
86         yIncrement = (int)Math.round( nodes[i].getBounds().getHeight() ) > yIncrement ? (int)Math.round(
87             nodes[i].getBounds().getHeight() ): yIncrement;
88     }
89     xIncrement += Integer.parseInt( getHorizontalSpacingField().getText() );
90     yIncrement += Integer.parseInt( getVerticalSpacingField().getText() );
91     int sideLength = Math.round( (float) Math.ceil( Math.sqrt( (double) nodes.length ) ) );
92     ArrayList list = new ArrayList();
93     for ( int i = 0 ; i < nodes.length ; i++ )
94     {
95         if ( (i % sideLength) == 0 )
96         {
97             xCoordinate = 0;
98         }
99         else
100         {
101             xCoordinate += xIncrement;
102         }
103         yCoordinate = (i / sideLength )* yIncrement;
104         list.add( new java.awt.Point( xCoordinate, yCoordinate ) );
105         // System.out.println( String.valueOf( xCoordinate )+ "-" + String.valueOf( yCoordinate ) );
106     }
107     return list;
108 }
109 /**
110 * Distribute a set of nodes at positions given in the second argument list.
111 * Creation date: (29-08-00 08:28:01)

```

```

110 * @param nodes edu.umd.cs.jazz.ZNode[]
111 * @param coordinates java.util.ArrayList
112 */
113 public void distributeCoordinates(edu.umd.cs.jazz.ZNode[] nodes, ArrayList coordinates)
114 {
115     // translate as many nodes as there are coordinate pairs and leave the remainder unchanged
116     int i,j;
117     for ( i = 0, j = 0 ; i < coordinates.size(); i++ )
118     {
119         if ( nodeLocked( nodes[ i ] )== false )
120         {
121             java.awt.Point coordinatePair = (java.awt.Point) coordinates.get(j);
122             setPosition( nodes[i], coordinatePair );
123             j++;
124         }
125     }
126 /**
127 * Insert the method's description here.
128 * Creation date: (12-10-00 23:50:36)
129 * @param nodes edu.umd.cs.jazz.ZNode[]
130 */
131 public void doLayout(edu.umd.cs.jazz.ZNode[] nodes)
132 {
133     // System.out.println("doLayout");
134     distributeCoordinates( nodes, computeCoordinates( nodes ) );
135 }
136 /**
137 * Insert the method's description here.
138 * Creation date: (12-10-00 11:56:01)
139 * @param node edu.umd.cs.jazz.ZGroup
140 */
141 public void doLayout(edu.umd.cs.jazz.ZGroup node)
142 {
143 }
144 /**
145 * Return the JPanel property value.
146 * @return javax.swing.JPanel
147 */
148 /* WARNING: THIS METHOD WILL BE REGENERATED. */
149 private javax.swing.JPanel getButtonPanel()
150 {
151     if (ivjButtonPanel == null) {
152         try {
153             ivjButtonPanel = new javax.swing.JPanel();
154             ivjButtonPanel.setName("ButtonPanel");
155             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
156             getButtonPanel().add(getOKButton(), getOKButton().getName());
157             getButtonPanel().add(getCancelButton(), getCancelButton().getName());
158             // user code begin {1}
159             // user code end
160         } catch (java.lang.Throwable ivjExc) {
161             // user code begin {2}
162             // user code end
163             handleException(ivjExc);
164         }
165     }
166     return ivjButtonPanel;
167 }
168 /**
169 * Return the JPanel property value.
170 * @return javax.swing.JPanel
171 */
172 /* WARNING: THIS METHOD WILL BE REGENERATED. */
173 private javax.swing.JPanel getButtonPanelFlowLayout()
174 {
175     javax.swing.JPanel ivjButtonPanelFlowLayout = null;
176     try {

```

```

176     /* Create part */
177     ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
178     ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
179 } catch (java.lang.Throwable ivjExc) {
180     handleException(ivjExc);
181 };
182 return ivjButtonPanelFlowLayout;
}
184 /**
 * Return the CancelButton property value.
185 * @return javax.swing.JButton
 */
186 /* WARNING: THIS METHOD WILL BE REGENERATED. */
187 private javax.swing.JButton getCancelButton() {
188     if (ivjCancelButton == null) {
189         try {
190             ivjCancelButton = new javax.swing.JButton();
191             ivjCancelButton.setName("CancelButton");
192             ivjCancelButton.setMnemonic('C');
193             ivjCancelButton.setText("Cancel");
194             // user code begin {1}
195             ivjCancelButton.addActionListener( new CancelHandler() );
196             // user code end
197         } catch (java.lang.Throwable ivjExc) {
198             // user code begin {2}
199             // user code end
200             handleException(ivjExc);
201         }
202     }
203     return ivjCancelButton;
}
204 }
205 /**
 * Return the DocumentHeightField property value.
 * @return javax.swing.JTextField
 */
206 /* WARNING: THIS METHOD WILL BE REGENERATED. */
207 private javax.swing.JTextField getDocumentHeightField() {
208     if (ivjDocumentHeightField == null) {
209         try {
210             ivjDocumentHeightField = new javax.swing.JTextField();
211             ivjDocumentHeightField.setName("DocumentHeightField");
212             // user code begin {1}
213             // user code end
214         } catch (java.lang.Throwable ivjExc) {
215             // user code begin {2}
216             // user code end
217             handleException(ivjExc);
218         }
219     }
220     return ivjDocumentHeightField;
}
221 }
222 /**
 * Return the DocumentHeightLabel property value.
 * @return javax.swing.JLabel
 */
223 /* WARNING: THIS METHOD WILL BE REGENERATED. */
224 private javax.swing.JLabel getDocumentHeightLabel() {
225     if (ivjDocumentHeightLabel == null) {
226         try {
227             ivjDocumentHeightLabel = new javax.swing.JLabel();
228             ivjDocumentHeightLabel.setName("DocumentHeightLabel");
229             ivjDocumentHeightLabel.setText("DocumentHeight:");
230             // user code begin {1}
231             // user code end
232         } catch (java.lang.Throwable ivjExc) {
233             // user code begin {2}

```

```

242         // user code end
243         handleException(ivjExc);
244     }
245     return ivjDocumentHeightLabel;
246 }
247 /**
248 * Return the dimensions of a document.
249 * Creation date: (06-09-00 00:46:03)
250 * @return java.awt.Dimension
251 */
252 public java.awt.Dimension getDocumentSize()
253 {
254     int height = getDocumentHeightField().getText().equals("") ?
255         getGRID_LAYOUT_DOCUMENT_HEIGHT_DEFAULT_VALUE() :
256             Integer.parseInt( getDocumentHeightField().getText() );
257     int width = getDocumentWidthField().getText().equals("") ?
258         getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() :
259             Integer.parseInt( getDocumentWidthField().getText() );
260     return new java.awt.Dimension( height, width );
261 }
262 /**
263 * Return the DocumentWidthField property value.
264 * @return javax.swing.JTextField
265 */
266 /* WARNING: THIS METHOD WILL BE REGENERATED. */
267 private javax.swing.JTextField getDocumentWidthField() {
268     if (ivjDocumentWidthField == null) {
269         try {
270             ivjDocumentWidthField = new javax.swing.JTextField();
271             ivjDocumentWidthField.setName("DocumentWidthField");
272             // user code begin {1}
273             // user code end
274         } catch (java.lang.Throwable ivjExc) {
275             // user code begin {2}
276             // user code end
277             handleException(ivjExc);
278         }
279     }
280     return ivjDocumentWidthField;
281 }
282 /**
283 * Return the DocumentWidthLabel property value.
284 * @return javax.swing.JLabel
285 */
286 /* WARNING: THIS METHOD WILL BE REGENERATED. */
287 private javax.swing.JLabel getDocumentWidthLabel() {
288     if (ivjDocumentWidthLabel == null) {
289         try {
290             ivjDocumentWidthLabel = new javax.swing.JLabel();
291             ivjDocumentWidthLabel.setName("DocumentWidthLabel");
292             ivjDocumentWidthLabel.setText("DocumentWidth:");
293             // user code begin {1}
294             // user code end
295         } catch (java.lang.Throwable ivjExc) {
296             // user code begin {2}
297             // user code end
298             handleException(ivjExc);
299         }
300     }
301     return ivjDocumentWidthLabel;
302 }
303 /**
304 * Insert the method's description here.
305 * Creation date: (12-10-00 16:01:25)
306 * @return java.lang.String

```

```

306  /*
307   * Insert the method's description here.
308   * Creation date: (12-10-00 16:18:11)
309   * @return int
310  */
311 /**
312  * Insert the method's description here.
313  * Creation date: (12-10-00 13:49:28)
314  * @return java.lang.String
315  */
316 public final static int getGRID_LAYOUT_DOCUMENT_HEIGHT() {
317     return GRID_LAYOUT_DOCUMENT_HEIGHT;
318 }
319 /**
320  * Insert the method's description here.
321  * Creation date: (12-10-00 14:35:05)
322  * @return int
323  */
324 public final static java.lang.String getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() {
325     return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING;
326 }
327 /**
328  * Insert the method's description here.
329  * Creation date: (12-10-00 13:49:59)
330  * @return int
331  */
332 public final static int getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE() {
333     return GRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING_DEFAULT_VALUE;
334 }
335 /**
336  * Insert the method's description here.
337  * Creation date: (12-10-00 14:34:25)
338  * @return int
339  */
340 public final static java.lang.String getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() {
341     return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING;
342 }
343 /**
344  * Insert the method's description here.
345  * Creation date: (12-10-00 16:01:47)
346  * @return java.lang.String
347  */
348 public final static int getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() {
349     return GRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE;
350 }
351 /**
352  * Insert the method's description here.
353  * Creation date: (12-10-00 16:17:06)
354  * @return int
355  */
356 public final static java.lang.String getGRID_LAYOUT_DOCUMENT_WIDTH() {
357     return GRID_LAYOUT_DOCUMENT_WIDTH;
358 }
359 /**
360  * Insert the method's description here.
361  * Creation date: (12-10-00 16:17:06)
362  * @return int
363  */
364 public final static int getGRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE() {
365     return GRID_LAYOUT_DOCUMENT_WIDTH_DEFAULT_VALUE;
366 }
367 /**
368  * Return the GridLayoutCustomizationBox property value.
369  * @return javax.swing.JDialog
370  */
371 /* WARNING: THIS METHOD WILL BE REGENERATED. */
372 private javax.swing.JDialog getGridLayoutCustomizationBox() {

```

```

372     if (ivjGridLayoutCustomizationBox == null) {
373         try {
374             ivjGridLayoutCustomizationBox = new javax.swing.JDialog();
375             ivjGridLayoutCustomizationBox.setName("GridLayoutCustomizationBox");
376             ivjGridLayoutCustomizationBox.setDefaultCloseOperation(javax.swing.WindowConstants.
377                 DISPOSE_ON_CLOSE);
378             ivjGridLayoutCustomizationBox.setBounds(24, 44, 398, 300);
379             ivjGridLayoutCustomizationBox.setModal(true);
380             ivjGridLayoutCustomizationBox.setTitle("GridLayoutCustomization");
381             getGridLayoutCustomizationBox().setContentPane(getJDialogContentPane());
382             // user code begin {1}
383             // user code end
384         } catch (java.lang.Throwable ivjExc) {
385             // user code begin {2}
386             // user code end
387             handleException(ivjExc);
388         }
389     }
390     return ivjGridLayoutCustomizationBox;
391 }
392 /**
393 * Return the HorizontalSpacingField property value.
394 * @return javax.swing.JTextField
395 */
396 /* WARNING: THIS METHOD WILL BE REGENERATED. */
397 private javax.swing.JTextField getHorizontalSpacingField() {
398     if (ivjHorizontalSpacingField == null) {
399         try {
400             ivjHorizontalSpacingField = new javax.swing.JTextField();
401             ivjHorizontalSpacingField.setName("HorizontalSpacingField");
402             // user code begin {1}
403             // user code end
404         } catch (java.lang.Throwable ivjExc) {
405             // user code begin {2}
406             // user code end
407             handleException(ivjExc);
408         }
409     }
410     return ivjHorizontalSpacingField;
411 }
412 /**
413 * Return the HorizontalSpacingLabel property value.
414 * @return javax.swing.JLabel
415 */
416 /* WARNING: THIS METHOD WILL BE REGENERATED. */
417 private javax.swing.JLabel getHorizontalSpacingLabel() {
418     if (ivjHorizontalSpacingLabel == null) {
419         try {
420             ivjHorizontalSpacingLabel = new javax.swing.JLabel();
421             ivjHorizontalSpacingLabel.setName("HorizontalSpacingLabel");
422             ivjHorizontalSpacingLabel.setText("HorizontalSpacing:");
423             // user code begin {1}
424             // user code end
425         } catch (java.lang.Throwable ivjExc) {
426             // user code begin {2}
427             // user code end
428             handleException(ivjExc);
429         }
430     }
431     return ivjHorizontalSpacingLabel;
432 }
433 /**
434 * Return the JDialogContentPane property value.
435 * @return javax.swing.JPanel
436 */
437 /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

    private javax.swing.JPanel getJDialogContentPane() {
438     if (ivjJDialogContentPane == null) {
        try {
440         ivjJDialogContentPane = new javax.swing.JPanel();
441         ivjJDialogContentPane.setName("JDialogContentPane");
442         ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
443         getJDialogContentPane().add(getMainPanel(), "Center");
444         getJDialogContentPane().add(getButtonPanel(), "South");
445         // user code begin {1}
446         // user code end
447     } catch (java.lang.Throwable ivjExc) {
448         // user code begin {2}
449         // user code end
450         handleException(ivjExc);
451     }
452   }
453   return ivjJDialogContentPane;
454 }
455 /**
456 * Return the MainPanel property value.
457 * @return javax.swing.JPanel
458 */
459 /* WARNING: THIS METHOD WILL BE REGENERATED. */
460 private javax.swing.JPanel getMainPanel() {
461     if (ivjMainPanel == null) {
462         try {
463             ivjMainPanel = new javax.swing.JPanel();
464             ivjMainPanel.setName("MainPanel");
465             ivjMainPanel.setLayout(new java.awt.GridBagLayout());
466
467             java.awt.GridBagConstraints constraintsDocumentWidthLabel = new java.awt.GridBagConstraints();
468             constraintsDocumentWidthLabel.gridx = 0; constraintsDocumentWidthLabel.gridy = 0;
469             constraintsDocumentWidthLabel.anchor = java.awt.GridBagConstraints.WEST;
470             constraintsDocumentWidthLabel.weighty = 1.0;
471             constraintsDocumentWidthLabel.insets = new java.awt.Insets(4, 4, 4, 4);
472             getMainPanel().add(getDocumentWidthLabel(), constraintsDocumentWidthLabel);
473
474             java.awt.GridBagConstraints constraintsDocumentHeightLabel = new java.awt.GridBagConstraints();
475             constraintsDocumentHeightLabel.gridx = 0; constraintsDocumentHeightLabel.gridy = 1;
476             constraintsDocumentHeightLabel.anchor = java.awt.GridBagConstraints.WEST;
477             constraintsDocumentHeightLabel.weighty = 1.0;
478             constraintsDocumentHeightLabel.insets = new java.awt.Insets(4, 4, 4, 4);
479             getMainPanel().add(getDocumentHeightLabel(), constraintsDocumentHeightLabel);
480
481             java.awt.GridBagConstraints constraintsHorizontalSpacingLabel = new java.awt.GridBagConstraints();
482             constraintsHorizontalSpacingLabel.gridx = 0; constraintsHorizontalSpacingLabel.gridy = 2;
483             constraintsHorizontalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
484             constraintsHorizontalSpacingLabel.weighty = 1.0;
485             constraintsHorizontalSpacingLabel.insets = new java.awt.Insets(4, 4, 4, 4);
486             getMainPanel().add(getHorizontalSpacingLabel(), constraintsHorizontalSpacingLabel);
487
488             java.awt.GridBagConstraints constraintsVerticalSpacingLabel = new java.awt.GridBagConstraints();
489             constraintsVerticalSpacingLabel.gridx = 0; constraintsVerticalSpacingLabel.gridy = 3;
490             constraintsVerticalSpacingLabel.anchor = java.awt.GridBagConstraints.WEST;
491             constraintsVerticalSpacingLabel.weighty = 1.0;
492             constraintsVerticalSpacingLabel.insets = new java.awt.Insets(4, 4, 4, 4);
493             getMainPanel().add(getVerticalSpacingLabel(), constraintsVerticalSpacingLabel);
494
495             java.awt.GridBagConstraints constraintsDocumentWidthField = new java.awt.GridBagConstraints();
496             constraintsDocumentWidthField.gridx = 1; constraintsDocumentWidthField.gridy = 0;
497             constraintsDocumentWidthField.fill = java.awt.GridBagConstraints.HORIZONTAL;
498             constraintsDocumentWidthField.anchor = java.awt.GridBagConstraints.EAST;
499             constraintsDocumentWidthField.weightx = 1.0;
500             constraintsDocumentWidthField.weighty = 1.0;
501             constraintsDocumentWidthField.insets = new java.awt.Insets(4, 4, 4, 4);
502
503         }
504     }
505   }

```

```

502     getMainPanel().add(getDocumentWidthField(), constraintsDocumentWidthField);
504
505     java.awt.GridBagConstraints constraintsDocumentHeightField = new java.awt.GridBagConstraints();
506     constraintsDocumentHeightField.gridx = 1; constraintsDocumentHeightField.gridy = 1;
507     constraintsDocumentHeightField.fill = java.awt.GridBagConstraints.HORIZONTAL;
508     constraintsDocumentHeightField.anchor = java.awt.GridBagConstraints.EAST;
509     constraintsDocumentHeightField.weightx = 1.0;
510     constraintsDocumentHeightField.weighty = 1.0;
511     constraintsDocumentHeightField.insets = new java.awt.Insets(4, 4, 4, 4);
512     getMainPanel().add(getDocumentHeightField(), constraintsDocumentHeightField);
513
514     java.awt.GridBagConstraints constraintsHorizontalSpacingField = new java.awt.GridBagConstraints()
515     ();
516     constraintsHorizontalSpacingField.gridx = 1; constraintsHorizontalSpacingField.gridy = 2;
517     constraintsHorizontalSpacingField.fill = java.awt.GridBagConstraints.HORIZONTAL;
518     constraintsHorizontalSpacingField.anchor = java.awt.GridBagConstraints.EAST;
519     constraintsHorizontalSpacingField.weightx = 1.0;
520     constraintsHorizontalSpacingField.weighty = 1.0;
521     constraintsHorizontalSpacingField.insets = new java.awt.Insets(4, 4, 4, 4);
522     getMainPanel().add(getHorizontalSpacingField(), constraintsHorizontalSpacingField);
523
524     java.awt.GridBagConstraints constraintsVerticalSpacingField = new java.awt.GridBagConstraints();
525     constraintsVerticalSpacingField.gridx = 1; constraintsVerticalSpacingField.gridy = 3;
526     constraintsVerticalSpacingField.fill = java.awt.GridBagConstraints.HORIZONTAL;
527     constraintsVerticalSpacingField.anchor = java.awt.GridBagConstraints.EAST;
528     constraintsVerticalSpacingField.weightx = 1.0;
529     constraintsVerticalSpacingField.weighty = 1.0;
530     constraintsVerticalSpacingField.insets = new java.awt.Insets(4, 4, 4, 4);
531     getMainPanel().add(getVerticalSpacingField(), constraintsVerticalSpacingField);
532     // user code begin {1}
533     ivjMainPanel.setBorder( ClientConstants.createDialogBorder( ivjMainPanel.getBackground() ) );
534     // user code end
535     } catch (java.lang.Throwable ivjExc) {
536         // user code begin {2}
537         // user code end
538         handleException(ivjExc);
539     }
540 }
541 /**
542 * Return the OKButton property value.
543 * @return javax.swing.JButton
544 */
545 /* WARNING: THIS METHOD WILL BE REGENERATED. */
546 private javax.swing.JButton getOKButton() {
547     if (ivjOKButton == null) {
548         try {
549             ivjOKButton = new javax.swing.JButton();
550             ivjOKButton.setName("OKButton");
551             ivjOKButton.setMnemonic('O');
552             ivjOKButton.setText("OK");
553             // user code begin {1}
554             ivjOKButton.addActionListener( new OKHandler() );
555             // user code end
556         } catch (java.lang.Throwable ivjExc) {
557             // user code begin {2}
558             // user code end
559             handleException(ivjExc);
560         }
561     }
562     return ivjOKButton;
563 }
564 /**
565 * Return the VerticalSpacingField property value.
566 * @return javax.swing.JTextField

```

```

      */
568 /* WARNING: THIS METHOD WILL BE REGENERATED. */
569 private javax.swing.JTextField getVerticalSpacingField() {
570     if (ivjVerticalSpacingField == null) {
571         try {
572             ivjVerticalSpacingField = new javax.swing.JTextField();
573             ivjVerticalSpacingField.setName("VerticalSpacingField");
574             // user code begin {1}
575             // user code end
576         } catch (java.lang.Throwable ivjExc) {
577             // user code begin {2}
578             // user code end
579             handleException(ivjExc);
580         }
581     }
582     return ivjVerticalSpacingField;
583 }
584 /**
585 * Return the VerticalSpacingLabel property value.
586 * @return javax.swing.JLabel
587 */
588 /* WARNING: THIS METHOD WILL BE REGENERATED. */
589 private javax.swing.JLabel getVerticalSpacingLabel() {
590     if (ivjVerticalSpacingLabel == null) {
591         try {
592             ivjVerticalSpacingLabel = new javax.swing.JLabel();
593             ivjVerticalSpacingLabel.setName("VerticalSpacingLabel");
594             ivjVerticalSpacingLabel.setText("Vertical_spacing:");
595             // user code begin {1}
596             // user code end
597         } catch (java.lang.Throwable ivjExc) {
598             // user code begin {2}
599             // user code end
600             handleException(ivjExc);
601         }
602     }
603     return ivjVerticalSpacingLabel;
604 }
605 /**
606 * Called whenever the part throws an exception.
607 * @param exception java.lang.Throwable
608 */
609 private void handleException(java.lang.Throwable exception) {
610     /* Uncomment the following lines to print uncaught exceptions to stdout */
611     // System.out.println("----- UNCAUGHT EXCEPTION -----");
612     // exception.printStackTrace(System.out);
613 }
614 /**
615 * Initialize the class.
616 */
617 /* WARNING: THIS METHOD WILL BE REGENERATED. */
618 private void initialize() {
619     try {
620         // user code begin {1}
621         // user code end
622     } catch (java.lang.Throwable ivjExc) {
623         handleException(ivjExc);
624     }
625     // user code begin {2}
626     // user code end
627 }
628 /**
629 * Insert the method's description here.
630 * Creation date: (12-10-00 15:31:05)
631 * @return boolean
632 */

```

```

        */
634 public boolean isCommitted() {
    return committed;
636 }
/** 
638 * main entrypoint - starts the part when it is run as an application
639 * @param args java.lang.String[]
640 */
641 public static void main(java.lang.String[] args) {
642     try {
643         GridPanoramaLayoutManager aGridPanoramaLayoutManager;
644         aGridPanoramaLayoutManager = new GridPanoramaLayoutManager();
645     } catch (Throwable exception) {
646         System.err.println("Exception occurred in main() of clientapp.ClientPanoramaLayout");
647         exception.printStackTrace(System.out);
648     }
649 }
650 /**
651 * Insert the method's description here.
652 * Creation date: (12-10-00 15:31:05)
653 * @param newCommitted boolean
654 */
655 public void setCommitted(boolean newCommitted) {
656     committed = newCommitted;
657 }
658 /**
659 * Set document specific preferences.
660 * Creation date: (12-10-00 13:46:17)
661 * @param newSettings java.util.Properties
662 */
663 public void setSettings(java.util.Properties newSettings)
664 {
665     if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() )!= null )
666     {
667         getDocumentHeightField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT() ) );
668     }
669     else
670     {
671         getDocumentHeightField().setText( String.valueOf( getDOCUMENT_HEIGHT_DEFAULT_VALUE() ) );
672     }
673     if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() )!= null )
674     {
675         getDocumentWidthField().setText( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_WIDTH() ) );
676     }
677     else
678     {
679         getDocumentWidthField().setText( String.valueOf( getDOCUMENT_WIDTH_DEFAULT_VALUE() ) );
680     }
681     if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() )!= null )
682     {
683         getHorizontalSpacingField().setText( newSettings.getProperty(
684             getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING() ) );
685     }
686     else
687     {
688         getHorizontalSpacingField().setText( String.valueOf(
689             getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING_DEFAULT_VALUE() ) );
690     }
691     if ( newSettings.getProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() )!= null )
692     {
693         getVerticalSpacingField().setText( newSettings.getProperty(
694             getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING() ) );
695     }
696     else
697     {

```

```

        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)
701 * @param owner javax.swing.JDialog
702 */
703 public void show(javax.swing.JDialog owner)
704 {
    getGridLayoutCustomizationBox().setLocationRelativeTo( owner );
705    getGridLayoutCustomizationBox().show();
}
706 /**
 * Take a properties table and update or insert (if non-existent) the <key,value> pairs defining the
 * configuration of this layout manager
707 * Creation date: (12-10-00 14:49:58)
708 * @return java.util.Properties
709 * @param currentSettings java.util.Properties
710 */
711 public java.util.Properties updateSettings(java.util.Properties currentSettings)
{
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HEIGHT(), getDocumentHeightField().getText() );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_WIDTH(), getDocumentWidthField().getText() );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_HORIZONTAL_SPACING(), getHorizontalSpacingField
        () );
    currentSettings.setProperty( getGRID_LAYOUT_DOCUMENT_VERTICAL_SPACING(), getVerticalSpacingField().
        getText() );
712    return currentSettings;
}
713 /**
 * Validate the contents of the document preferences configuration box.
714 * Creation date: (06-09-00 13:55:19)
715 * @return boolean
716 */
717 public boolean validateFields()
{
    peerviewmisc.MessageBox messageBox = new peerviewmisc.MessageBox();
    boolean valid = true;

718    if ( ( Integer.parseInt( getDocumentHeightField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUT_DOCUMENT_HEIGHT() )
        || ( Integer.parseInt( getDocumentHeightField().getText() ) < ClientConstants.
            getMIN_GRID_LAYOUT_DOCUMENT_HEIGHT() ) )
719    {
        messageBox.setText( ClientConstants.getDOCUMENT_SIZE_OUT_OF_BOUNDS() );
720        valid = false;
    }
721    else if ( ( Integer.parseInt( getDocumentWidthField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUT_DOCUMENT_WIDTH() )
        || ( Integer.parseInt( getDocumentWidthField().getText() ) < ClientConstants.
            getMIN_GRID_LAYOUT_DOCUMENT_WIDTH() ) )
722    {
        messageBox.setText( ClientConstants.getDOCUMENT_SIZE_OUT_OF_BOUNDS() );
723        valid = false;
    }
724    else if ( ( Integer.parseInt( getHorizontalSpacingField().getText() ) > ClientConstants.
        getMAX_GRID_LAYOUT_HORIZONTAL_SPACING() )
        || ( Integer.parseInt( getHorizontalSpacingField().getText() ) < ClientConstants.
            getMIN_GRID_LAYOUT_HORIZONTAL_SPACING() ) )
725    {
        messageBox.setText( ClientConstants.getGRID_LAYOUT_SPACING_OUT_OF_BOUNDS() );
726        valid = false;
    }
}

```

```

        else if ( ( Integer.parseInt( getVerticalSpacingField().getText() ) > ClientConstants.
            getMAX_GRID_LAYOUT_VERTICAL_SPACING() )
    ||   ( 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.jsdst.*;
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();
39     }
42 }
44
    public class DeleteGroupButtonHandler implements ActionListener
46 {
        public void actionPerformed ( ActionEvent ae )
47 {
            // System.out.println( "ramte deletegroupbuttonhandler" );
48
}
}

```

```

        deleteGroup();
50    }
}
52
public class JoinGroupButtonHandler implements ActionListener
{
    public void actionPerformed ( ActionEvent ae )
56    {
        joinGroup();
58    }
}
60
public class CloseButtonHandler implements ActionListener
62{
    public void actionPerformed( ActionEvent ae )
64    {
        dispose();
66    }
}
68 private javax.swing.JButton ivjCloseButton = null;
private javax.swing.JButton ivjCreateGroupButton = null;
70 private javax.swing.JButton ivjDeleteGroupButton = null;
private javax.swing.JPanel ivjJDialogContentPane = null;
72 private javax.swing.JButton ivjJoinGroupButton = null;
private javax.swing.JPanel ivj JPanel2 = null;
74 private java.awt.FlowLayout ivj JPanel2FlowLayout = null;
private javax.swing.JPanel ivj JPanel3 = null;
76 private java.awt.FlowLayout ivj JPanel3FlowLayout = null;
private javax.swing.JScrollPane ivj JScrollPane1 = null;
78 private javax.swing.JTable ivjScrollPaneTable = null;
private DeleteGroup ivjDeleteGroup1 = null;
80 private NewGroup ivjNewGroup1 = null;
private peerviewmisc.WorkGroup workGroup = null;
82 private EditGroup ivjEditGroup1 = null;
private javax.swing.JButton ivjEditGroupButton = null;
84 private javax.swing.table.DefaultTableModel tableModel;
private javax.swing.JPanel ivj JPanelPanel = null;
86 private javax.swing.JPanel ivj JPanelPanel = null;
private peerviewmisc.WorkGroup[] groupDirectoryValues = null;
88 IvjEventHandler ivjEventHandler = new IvjEventHandler();
private ClientApplication clientApplication = null;
90
class IvjEventHandler implements java.awt.event.ActionListener {
92    public void actionPerformed(java.awt.event.ActionEvent e) {
        if (e.getSource() == GroupDirectory.this.getJoinGroupButton())
94        connEtoM1(e);
        if (e.getSource() == GroupDirectory.this.getCreateGroupButton())
96        connEtoM2(e);
        if (e.getSource() == GroupDirectory.this.getEditGroupButton())
98        connEtoM3(e);
        if (e.getSource() == GroupDirectory.this.getDeleteGroupButton())
100       connEtoM4(e);
        if (e.getSource() == GroupDirectory.this.getCloseButton())
102       connEtoM5(e);
    };
104   };
}
106 /**
 * GroupDirectory constructor comment.
 */
108 public GroupDirectory() {
    super();
110   initialize();
}
112 /**
 * GroupDirectory constructor comment.
 */
114 * @param owner java.awt.Dialog

```

```

    */
116 public GroupDirectory(java.awt.Dialog owner) {
    super(owner);
118 }
 /**
120 * GroupDirectory constructor comment.
121 * @param owner java.awt.Dialog
122 * @param title java.lang.String
123 */
124 public GroupDirectory(java.awt.Dialog owner, String title) {
    super(owner, title);
126 }
 /**
128 * GroupDirectory constructor comment.
129 * @param owner java.awt.Dialog
130 * @param title java.lang.String
131 * @param modal boolean
132 */
133 public GroupDirectory(java.awt.Dialog owner, String title, boolean modal) {
134     super(owner, title, modal);
135 }
136 /**
137 * GroupDirectory constructor comment.
138 * @param owner java.awt.Dialog
139 * @param modal boolean
140 */
141 public GroupDirectory(java.awt.Dialog owner, boolean modal) {
142     super(owner, modal);
143 }
144 /**
145 * GroupDirectory constructor comment.
146 * @param owner java.awt.Frame
147 */
148 public GroupDirectory(java.awt.Frame owner) {
    super(owner);
149 }
150 /**
151 * GroupDirectory constructor comment.
152 * @param owner java.awt.Frame
153 * @param title java.lang.String
154 */
155 public GroupDirectory(java.awt.Frame owner, String title) {
    super(owner, title);
156 }
157 /**
158 * GroupDirectory constructor comment.
159 * @param owner java.awt.Frame
160 * @param title java.lang.String
161 * @param modal boolean
162 */
163 public GroupDirectory(java.awt.Frame owner, String title, boolean modal) {
164     super(owner, title, modal);
165 }
166 /**
167 * GroupDirectory constructor comment.
168 * @param owner java.awt.Frame
169 * @param modal boolean
170 */
171 public GroupDirectory(java.awt.Frame owner, boolean modal) {
172     super(owner, modal);
173 }
174 /**
175 * connEtoM1: (JoinGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
176 *           joinGroup()V)
177 * @param arg1 java.awt.event.ActionEvent
178 */

```

```

180 /* WARNING: THIS METHOD WILL BE REGENERATED. */
181 private void connEtoM1(java.awt.event.ActionEvent arg1) {
182     try {
183         // user code begin {1}
184         // user code end
185         this.joinGroup();
186         // user code begin {2}
187         // user code end
188     } catch (java.lang.Throwable ivjExc) {
189         // user code begin {3}
190         // user code end
191         handleException(ivjExc);
192     }
193 }
194 /**
195 * connEtoM2: (CreateGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
196 *      createNewGroup()V)
197 * @param arg1 java.awt.event.ActionEvent
198 */
199 /* WARNING: THIS METHOD WILL BE REGENERATED. */
200 private void connEtoM2(java.awt.event.ActionEvent arg1) {
201     try {
202         // user code begin {1}
203         // user code end
204         this.createNewGroup();
205         // user code begin {2}
206         // user code end
207     } catch (java.lang.Throwable ivjExc) {
208         // user code begin {3}
209         // user code end
210         handleException(ivjExc);
211     }
212 }
213 /**
214 * connEtoM3: (EditGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
215 *      editExistingGroup()V)
216 * @param arg1 java.awt.event.ActionEvent
217 */
218 /* WARNING: THIS METHOD WILL BE REGENERATED. */
219 private void connEtoM3(java.awt.event.ActionEvent arg1) {
220     try {
221         // user code begin {1}
222         // System.out.println( "FYR" );
223         // user code end
224         this.editExistingGroup();
225         // user code begin {2}
226         // user code end
227     } catch (java.lang.Throwable ivjExc) {
228         // user code begin {3}
229         // user code end
230         handleException(ivjExc);
231     }
232 }
233 /**
234 * connEtoM4: (DeleteGroupButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.
235 *      deleteGroup()V)
236 * @param arg1 java.awt.event.ActionEvent
237 */
238 /* WARNING: THIS METHOD WILL BE REGENERATED. */
239 private void connEtoM4(java.awt.event.ActionEvent arg1) {
240     try {
241         // user code begin {1}
242         // user code end
243         this.deleteGroup();
244         // user code begin {2}
245         // user code end

```

```

    } catch (java.lang.Throwable ivjExc) {
244     // user code begin {3}
245     // user code end
246     handleException(ivjExc);
    }
248 }
249 /**
250 * connEtoM5: (CloseButton.action.actionPerformed(java.awt.event.ActionEvent) --> GroupDirectory.dispose
251     ()V)
252 * @param arg1 java.awt.event.ActionEvent
253 */
254 /* WARNING: THIS METHOD WILL BE REGENERATED. */
255 private void connEtoM5(java.awt.event.ActionEvent arg1) {
    try {
256     // user code begin {1}
257     // user code end
258     this.dispose();
259     // user code begin {2}
260     // user code end
261     } catch (java.lang.Throwable ivjExc) {
262     // user code begin {3}
263     // user code end
264     handleException(ivjExc);
    }
266 }
267 /**
268 * Display a dialog box for entering information about a new group and create a corresponding object if
269     the user acknowledges
270     * his or her entries.
271 * Creation date: (03-07-00 21:44:35)
272 */
273 public void createNewGroup()
{
274     NewGroup newGroup = getNewGroup1();

275     newGroup.setWorkGroup( new WorkGroup() );
276     newGroup.show();
277     WorkGroup workGroup = newGroup.getWorkGroup();
278     // if the workgroup box wasn't cancelled
279     if ( newGroup.isIsDiscarded() == false )
    {
280         // complete the non-editable fields of the new workgroup
281         workGroup.setCreationDate( new java.util.Date() );
282         workGroup.setCreator( (String) ClientInfo.getPreferences().getProperty( ClientConstants.
283             AUTHOR_NAME() );
284         getClientApplication().getClientManager().createWorkGroup( workGroup );
285     }
286 }
287 /**
288 * Delete whatever group is currently selected in the group directory listing if the user acknowledges
289     the deletion.
290 * Creation date: (03-07-00 21:45:09)
291 */
292 public void deleteGroup()
{
293     int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
294     // if no selection
295     if ( selectedRow == -1 )
    {
296         MessageBox messageBox = new MessageBox();
297         messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_DELETE() );
298         messageBox.show();
    }
299     else
300     {
301         WorkGroup selectedGroup = (WorkGroup) groupDirectoryValues[ selectedRow ];
302         // if the selected group has active participants
303     }
304 }

```

```

306     if ( selectedGroup.getParticipants().size() > 0 )
307     {
308         MessageBox messageBox = new MessageBox();
309         messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_DELETED_DUE_TO_ACTIVE_PARTICIPANTS() );
310         messageBox.show();
311         return;
312     }
313     ConfirmationBox confirmationBox = new ConfirmationBox();
314     confirmationBox.setText( ClientConstants.getCONFIRM_WORKGROUP_DELETION( selectedGroup.getGroupName
            () ) );
315     confirmationBox.show();
316     if ( confirmationBox.isConfirmed() == true )
317     {
318         getClientApplication().getClientManager().removeWorkGroup( selectedGroup );
319     }
320 }
321 /**
322 * Disable all buttons that trigger actions involving communication with the server. Used in case of an
323 * inactive server
324 * or a broken communications channel.
325 * Creation date: (10-08-00 16:13:10)
326 */
327 private void disableButtons()
328 {
329     getJoinGroupButton().setEnabled( false );
330     getCreateGroupButton().setEnabled( false );
331     getEditGroupButton().setEnabled( false );
332     getDeleteGroupButton().setEnabled( false );
333 }
334 /**
335 * Display an edit box containing the attributes of the group currently selected in the group directory.
336 * Creation date: (03-07-00 21:44:52)
337 */
338 public void editExistingGroup()
339 {
340     int selectedRow = getScrollPaneTable().getSelectionModel().getMinSelectionIndex();
341     // if no selection
342     if ( selectedRow == -1 )
343     {
344         MessageBox messageBox = new MessageBox();
345         messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_EDIT_MESSAGE() );
346         messageBox.show();
347     }
348     else
349     {
350         WorkGroup backup = new WorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
351         // if the selected group has active participants
352         if ( backup.getParticipants().size() > 0 )
353         {
354             MessageBox messageBox = new MessageBox();
355             messageBox.setText( ClientConstants.getGROUP_CANNOT_BE_EDITED_DUE_TO_ACTIVE_PARTICIPANTS() );
356             messageBox.show();
357             return;
358         }
359         getEditGroup1().setWorkGroup( (WorkGroup) groupDirectoryValues[ selectedRow ] );
360         getEditGroup1().show();
361         if ( getEditGroup1().isIsDiscarded() == false )
362         {
363             getClientApplication().getClientManager().removeWorkGroup( backup );
364             WorkGroup newWorkGroup = getEditGroup1().getWorkGroup();
365             newWorkGroup.setCreationDate( new java.util.Date() );
366             newWorkGroup.setCreator( ClientInfo.getPreferences().getProperty( ClientConstants.getAUTHOR_NAME
            () ) );

```

```

        // the below automatically causes the server to broadcast a group directory update to all
        // clients, thus
368    // eliminating the need to update the locally cached copy in the clientInfo object.
369    getClientApplication().getClientManager().createWorkGroup( newWorkGroup );
370}
371}
372}
373/**
374 * Insert the method's description here.
375 * Creation date: (10-08-00 16:13:27)
376 */
377private void enableButtons()
378{
379    getJoinGroupButton().setEnabled( true );
380    getCreateGroupButton().setEnabled( true );
381    getEditGroupButton().setEnabled( true );
382    getDeleteGroupButton().setEnabled( true );
383}
384}
385/**
386 * Return the JPanel1 property value.
387 * @return javax.swing.JPanel
388 */
389/* WARNING: THIS METHOD WILL BE REGENERATED. */
390private javax.swing.JPanel getButtonPanel() {
391    if (ivjButtonPanel == null) {
392        try {
393            ivjButtonPanel = new javax.swing.JPanel();
394            ivjButtonPanel.setName("ButtonPanel");
395            ivjButtonPanel.setLayout(new java.awt.BorderLayout());
396            getButtonPanel().add(getJPanel3(), "East");
397            getButtonPanel().add(getJPanel2(), "West");
398            // user code begin {1}
399            getButtonPanel().setBorder( javax.swing.BorderFactory.createMatteBorder(
400                ClientConstants.getBORDER_WIDTH(),
401                ClientConstants.getBORDER_WIDTH(),
402                ClientConstants.getBORDER_WIDTH(),
403                ClientConstants.getBORDER_WIDTH(),
404                getButtonPanel().getBackground() ) );
405            // user code end
406        } catch (java.lang.Throwable ivjExc) {
407            // user code begin {2}
408            // user code end
409            handleException(ivjExc);
410        }
411    }
412    return ivjButtonPanel;
413}
414/**
415 * Insert the method's description here.
416 * Creation date: (10-08-00 15:41:17)
417 * @return clientapp.ClientApplication
418 */
419public ClientApplication getClientApplication() {
420    return clientApplication;
421}
422/**
423 * Return the CloseButton property value.
424 * @return javax.swing.JButton
425 */
426/* WARNING: THIS METHOD WILL BE REGENERATED. */
427private javax.swing.JButton getCloseButton() {
428    if (ivjCloseButton == null) {
429        try {
430            ivjCloseButton = new javax.swing.JButton();
431            ivjCloseButton.setName("CloseButton");

```

```

432         ivjCloseButton.setMnemonic('c');
433         ivjCloseButton.setText("Close");
434         // user code begin {1}
435         ivjCloseButton.addActionListener( new CloseButtonHandler() );
436         // user code end
437     } catch (java.lang.Throwable ivjExc) {
438         // user code begin {2}
439         // user code end
440         handleException(ivjExc);
441     }
442 }
443     return ivjCloseButton;
444 }
445 /**
446 * Return the CreateGroupButton property value.
447 * @return javax.swing.JButton
448 */
449 /* WARNING: THIS METHOD WILL BE REGENERATED. */
450 private javax.swing.JButton getCreateGroupButton() {
451     if (ivjCreateGroupButton == null) {
452         try {
453             ivjCreateGroupButton = new javax.swing.JButton();
454             ivjCreateGroupButton.setName("CreateGroupButton");
455             ivjCreateGroupButton.setMnemonic('g');
456             ivjCreateGroupButton.setText("CreateGroup");
457             // user code begin {1}
458             //ivjCreateGroupButton.addActionListener( new CreateGroupButtonHandler() );
459             // user code end
460         } catch (java.lang.Throwable ivjExc) {
461             // user code begin {2}
462             // user code end
463             handleException(ivjExc);
464         }
465     }
466     return ivjCreateGroupButton;
467 }
468 /**
469 * Return the DeleteGroup1 property value.
470 * @return peerviewmisc.DeleteGroup
471 */
472 /* WARNING: THIS METHOD WILL BE REGENERATED. */
473 public peerviewmisc.DeleteGroup getDeleteGroup1() {
474     if (ivjDeleteGroup1 == null) {
475         try {
476             ivjDeleteGroup1 = new peerviewmisc.DeleteGroup();
477             ivjDeleteGroup1.setName("DeleteGroup1");
478             // user code begin {1}
479             // user code end
480         } catch (java.lang.Throwable ivjExc) {
481             // user code begin {2}
482             // user code end
483             handleException(ivjExc);
484         }
485     }
486     return ivjDeleteGroup1;
487 }
488 /**
489 * Return the DeleteGroupButton property value.
490 * @return javax.swing.JButton
491 */
492 /* WARNING: THIS METHOD WILL BE REGENERATED. */
493 private javax.swing.JButton getDeleteGroupButton() {
494     if (ivjDeleteGroupButton == null) {
495         try {
496             ivjDeleteGroupButton = new javax.swing.JButton();
497             ivjDeleteGroupButton.setName("DeleteGroupButton");

```

```

498         ivjDeleteGroupButton.setMnemonic('D');
499         ivjDeleteGroupButton.setText("Delete_Ugroup");
500         // user code begin {1}
501         //ivjDeleteGroupButton.addActionListener( new DeleteGroupButtonHandler() );
502         // user code end
503     } catch (java.lang.Throwable ivjExc) {
504         // user code begin {2}
505         // user code end
506         handleException(ivjExc);
507     }
508 }
509 return ivjDeleteGroupButton;
510 }
511 /**
512 * Return the EditGroup1 property value.
513 * @return peerviewmisc.EditGroup
514 */
515 /* WARNING: THIS METHOD WILL BE REGENERATED. */
516 public peerviewmisc.EditGroup getEditGroup1() {
517     if (ivjEditGroup1 == null) {
518         try {
519             ivjEditGroup1 = new peerviewmisc.EditGroup();
520             ivjEditGroup1.setName("EditGroup1");
521             ivjEditGroup1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
522             // user code begin {1}
523             ivjEditGroup1.setGroupDirectory( this );
524             // user code end
525         } catch (java.lang.Throwable ivjExc) {
526             // user code begin {2}
527             // user code end
528             handleException(ivjExc);
529         }
530     }
531     return ivjEditGroup1;
532 }
533 /**
534 * Return the EditGroupButton property value.
535 * @return javax.swing.JButton
536 */
537 /* WARNING: THIS METHOD WILL BE REGENERATED. */
538 public javax.swing.JButton getEditGroupButton() {
539     if (ivjEditGroupButton == null) {
540         try {
541             ivjEditGroupButton = new javax.swing.JButton();
542             ivjEditGroupButton.setName("EditGroupButton");
543             ivjEditGroupButton.setMnemonic('E');
544             ivjEditGroupButton.setText("Edit_Ugroup");
545             // user code begin {1}
546             //ivjEditGroupButton.addActionListener( new EditGroupButtonHandler() );
547             // user code end
548         } catch (java.lang.Throwable ivjExc) {
549             // user code begin {2}
550             // user code end
551             handleException(ivjExc);
552         }
553     }
554     return ivjEditGroupButton;
555 }
556 /**
557 * Insert the method's description here.
558 * Creation date: (26-07-00 21:09:59)
559 * @return peerviewmisc.WorkGroup[]
560 */
561 public peerviewmisc.WorkGroup[] getGroupDirectoryValues() {
562     return groupDirectoryValues;
563 }

```

```

564 /**
565  * Return the JDialogContentPane property value.
566  * @return javax.swing.JPanel
567  */
568 /* WARNING: THIS METHOD WILL BE REGENERATED. */
569 private javax.swing.JPanel getJDialogContentPane() {
570     if (ivjJDialogContentPane == null) {
571         try {
572             ivjJDialogContentPane = new javax.swing.JPanel();
573             ivjJDialogContentPane.setName("JDialogContentPane");
574             ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
575             getJDialogContentPane().add(getTablePanel(), "Center");
576             getJDialogContentPane().add(getButtonPanel(), "South");
577             // user code begin {1}
578             // user code end
579         } catch (java.lang.Throwable ivjExc) {
580             // user code begin {2}
581             // user code end
582             handleException(ivjExc);
583         }
584     }
585     return ivjJDialogContentPane;
586 }
587 /**
588  * Return the JoinGroupButton property value.
589  * @return javax.swing.JButton
590  */
591 /* WARNING: THIS METHOD WILL BE REGENERATED. */
592 private javax.swing.JButton getJoinGroupButton() {
593     if (ivjJoinGroupButton == null) {
594         try {
595             ivjJoinGroupButton = new javax.swing.JButton();
596             ivjJoinGroupButton.setName("JoinGroupButton");
597             ivjJoinGroupButton.setMnemonic('J');
598             ivjJoinGroupButton.setText("JoinGroup");
599             // user code begin {1}
600             //ivjJoinGroupButton.addActionListener( new JoinGroupButtonHandler() );
601             // user code end
602         } catch (java.lang.Throwable ivjExc) {
603             // user code begin {2}
604             // user code end
605             handleException(ivjExc);
606         }
607     }
608     return ivjJoinGroupButton;
609 }
610 /**
611  * Return the JPanel2 property value.
612  * @return javax.swing.JPanel
613  */
614 /* WARNING: THIS METHOD WILL BE REGENERATED. */
615 private javax.swing.JPanel getJPanel2() {
616     if (ivjJPanel2 == null) {
617         try {
618             ivjJPanel2 = new javax.swing.JPanel();
619             ivjJPanel2.setName("JPanel2");
620             ivjJPanel2.setLayout(getJPanel2FlowLayout());
621             getJPanel2().add(getJoinGroupButton(), getJoinGroupButton().getName());
622             getJPanel2().add(getCreateGroupButton(), getCreateGroupButton().getName());
623             ivjJPanel2.add(getEditGroupButton());
624             ivjJPanel2.add(getDeleteGroupButton());
625             // user code begin {1}
626             // user code end
627         } catch (java.lang.Throwable ivjExc) {
628             // user code begin {2}
629             // user code end

```

```

630         handleException(ivjExc);
631     }
632   }
633   return ivj JPanel2;
634 }
635 /**
636  * Return the JPanel2FlowLayout property value.
637  * @return java.awt.FlowLayout
638  */
639 /* WARNING: THIS METHOD WILL BE REGENERATED. */
640 private java.awt.FlowLayout get JPanel2FlowLayout() {
641   java.awt.FlowLayout ivj JPanel2FlowLayout = null;
642   try {
643     /* Create part */
644     ivj JPanel2FlowLayout = new java.awt.FlowLayout();
645     ivj JPanel2FlowLayout.setAlignment(java.awt.FlowLayout.LEFT);
646   } catch (java.lang.Throwable ivjExc) {
647     handleException(ivjExc);
648   };
649   return ivj JPanel2FlowLayout;
650 }
651 /**
652  * Return the JPanel3 property value.
653  * @return javax.swing.JPanel
654  */
655 /* WARNING: THIS METHOD WILL BE REGENERATED. */
656 private javax.swing.JPanel get JPanel3() {
657   if (ivj JPanel3 == null) {
658     try {
659       ivj JPanel3 = new javax.swing.JPanel();
660       ivj JPanel3.setName(" JPanel3");
661       ivj JPanel3.setLayout(get JPanel3FlowLayout());
662       get JPanel3().add(get closeButton(), get closeButton().getName());
663       // user code begin {1}
664       // user code end
665     } catch (java.lang.Throwable ivjExc) {
666       // user code begin {2}
667       // user code end
668       handleException(ivjExc);
669     }
670   }
671   return ivj JPanel3;
672 }
673 /**
674  * Return the JPanel3FlowLayout property value.
675  * @return java.awt.FlowLayout
676  */
677 /* WARNING: THIS METHOD WILL BE REGENERATED. */
678 private java.awt.FlowLayout get JPanel3FlowLayout() {
679   java.awt.FlowLayout ivj JPanel3FlowLayout = null;
680   try {
681     /* Create part */
682     ivj JPanel3FlowLayout = new java.awt.FlowLayout();
683     ivj JPanel3FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
684   } catch (java.lang.Throwable ivjExc) {
685     handleException(ivjExc);
686   };
687   return ivj JPanel3FlowLayout;
688 }
689 /**
690  * Return the JScrollPane1 property value.
691  * @return javax.swing.JScrollPane
692  */
693 /* WARNING: THIS METHOD WILL BE REGENERATED. */
694 private javax.swing.JScrollPane get JScrollPane1() {
695   if (ivj JScrollPane1 == null) {

```

```

696     try {
697         ivjJScrollPane1 = new javax.swing.JScrollPane();
698         ivjJScrollPane1.setName("JScrollPane");
699         ivjJScrollPane1.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
700         ivjJScrollPane1.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
701             );
702         ivjJScrollPane1.setDoubleBuffered(true);
703         getJScrollPane1().setViewportView(getScrollPaneTable());
704         // user code begin {1}
705         // user code end
706     } catch (java.lang.Throwable ivjExc) {
707         // user code begin {2}
708         // user code end
709         handleException(ivjExc);
710     }
711 }
712 */
713 * Return the NewGroup1 property value.
714 * @return peerviewmisc.NewGroup
715 */
716 /* WARNING: THIS METHOD WILL BE REGENERATED. */
717 public peerviewmisc.NewGroup getNewGroup1() {
718     if (ivjNewGroup1 == null) {
719         try {
720             ivjNewGroup1 = new peerviewmisc.NewGroup();
721             ivjNewGroup1.setName("NewGroup1");
722             // user code begin {1}
723             ivjNewGroup1.setGroupDirectory( this );
724             // user code end
725         } catch (java.lang.Throwable ivjExc) {
726             // user code begin {2}
727             // user code end
728             handleException(ivjExc);
729         }
730     }
731     return ivjNewGroup1;
732 }
733 */
734 * Return the ScrollPaneTable property value.
735 * @return javax.swing.JTable
736 */
737 /* WARNING: THIS METHOD WILL BE REGENERATED. */
738 private javax.swing.JTable getScrollPaneTable() {
739     if (ivjScrollPaneTable == null) {
740         try {
741             ivjScrollPaneTable = new javax.swing.JTable();
742             ivjScrollPaneTable.setName("ScrollPaneTable");
743             getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
744             getJScrollPane1().getViewport().setBackingStoreEnabled(true);
745             ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
746             ivjScrollPaneTable.setPreferredSize(new java.awt.Dimension(716,412));
747             ivjScrollPaneTable.setBounds(0, 0, 716, 412);
748             ivjScrollPaneTable.setPreferredScrollableViewportSize(new java.awt.Dimension(700, 400));
749             ivjScrollPaneTable.setAutoCreateColumnsFromModel(true);
750             // user code begin {1}
751             ivjScrollPaneTable.setSelectionMode( javax.swing.ListSelectionModel.SINGLE_SELECTION );
752             ivjScrollPaneTable.setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
753             // user code end
754         } catch (java.lang.Throwable ivjExc) {
755             // user code begin {2}
756             // user code end
757             handleException(ivjExc);
758         }
759     }
760 }

```

```

        return ivjScrollPaneTable;
762    }
763    /**
764     * Insert the method's description here.
765     * Creation date: (04-07-00 19:38:32)
766     * @return javax.swing.table.DefaultTableModel
767     */
768    public javax.swing.table.DefaultTableModel getTableModel() {
769        return tableModel;
770    }
771    /**
772     * Return the JPanel4 property value.
773     * @return javax.swing.JPanel
774     */
775    /* WARNING: THIS METHOD WILL BE REGENERATED. */
776    private javax.swing.JPanel getTablePanel() {
777        if (ivjTablePanel == null) {
778            try {
779                ivjTablePanel = new javax.swing.JPanel();
780                ivjTablePanel.setName("TablePanel");
781                ivjTablePanel.setLayout(new java.awt.BorderLayout());
782                getTablePanel().add(getJScrollPane1(), "Center");
783                // user code begin {1}
784                getTablePanel().setBorder( ClientConstants.createDialogBorder( getTablePanel().getBackground() )
785                    );
786                // user code end
787            } catch (java.lang.Throwable ivjExc) {
788                // user code begin {2}
789                // user code end
790                handleException(ivjExc);
791            }
792        }
793        return ivjTablePanel;
794    }
795    /**
796     * Insert the method's description here.
797     * Creation date: (03-07-00 21:00:14)
798     * @return peerviewmisc.WorkGroup
799     */
800    public peerviewmisc.WorkGroup getWorkGroup() {
801        return workGroup;
802    }
803    /**
804     * Traverses the group directory (list of active groups) to determine if any group matches the group
805     * name and description parameters
806     * Creation date: (26-07-00 16:59:38)
807     * @return boolean
808     */
809    public boolean groupExists(String groupName, String description)
810    {
811        boolean exists = false;
812        for ( int i = 0; i < groupDirectoryValues.length; i++ )
813        {
814            if ( ( (WorkGroup) groupDirectoryValues[i]).getGroupName().equalsIgnoreCase( groupName )&&
815                ( (WorkGroup) groupDirectoryValues[i]).getDescription().equalsIgnoreCase( description ) )
816            {
817                exists = true;
818                break;
819            }
820        }
821        return exists;
822    }
823    /**
824     * Called whenever the part throws an exception.

```

```

826     * @param exception java.lang.Throwable
827     */
828     private void handleException(java.lang.Throwable exception) {
829
830         /* Uncomment the following lines to print uncaught exceptions to stdout */
831         // System.out.println("----- UNCAUGHT EXCEPTION -----");
832         // exception.printStackTrace(System.out);
833     }
834     /**
835      * Initializes connections
836      * @exception java.lang.Exception The exception description.
837      */
838     /* WARNING: THIS METHOD WILL BE REGENERATED. */
839     private void initConnections() throws java.lang.Exception {
840         // user code begin {1}
841         // user code end
842         getJoinGroupButton().addActionListener(ivjEventHandler);
843         getCreateGroupButton().addActionListener(ivjEventHandler);
844         getEditGroupButton().addActionListener(ivjEventHandler);
845         getDeleteGroupButton().addActionListener(ivjEventHandler);
846         getCloseButton().addActionListener(ivjEventHandler);
847     }
848     /**
849      * Initialize the class.
850      */
851     /* WARNING: THIS METHOD WILL BE REGENERATED. */
852     private void initialize() {
853         try {
854             // user code begin {1}
855             // user code end
856             setName("GroupDirectory");
857             setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
858             setSize(845, 467);
859             setModal(true);
860             setTitle("Group_directory");
861             setContentPane(getJDialogContentPane());
862             initConnections();
863         } catch (java.lang.Throwable ivjExc) {
864             handleException(ivjExc);
865         }
866         // user code begin {2}
867         getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS );
868         setLocationRelativeTo( getOwner() );
869         addWindowListener( new GroupDirectoryWindowListener() );
870         // user code end
871     }
872     /**
873      * Insert the method's description here.
874      * Creation date: (04-07-00 23:11:34)
875      */
876     public void initTable()
877     {
878         WorkGroup workGroup = new WorkGroup();
879         tableModel = new javax.swing.table.DefaultTableModel();
880         ivjScrollPaneTable.setModel( tableModel );
881
882         String columnNames[] = WorkGroup.getFieldNames();
883         for ( int i = 0 ; i < columnNames.length; i++ )
884         {
885             tableModel.addColumn( columnNames[i] );
886         }
887     }
888     /**
889      * Insert the method's description here.
890      * Creation date: (16-08-00 18:15:00)
891      */

```

```

public void initWindow()
892 {
893 }
894 /**
895 * Insert the method's description here.
896 * Creation date: (06-07-00 12:48:00)
897 */
898 public void joinGroup()
{
899     int selectedIndex = getScrollPaneTable().getSelectedRow();
900     // if no row is selected
901     if ( selectedIndex == -1 )
902     {
903         MessageBox messageBox = new MessageBox();
904         messageBox.setText( ClientConstants.getNO_GROUP_SELECTED_TO_JOIN_MESSAGE() );
905         messageBox.show();
906         return;
907     }
908     WorkGroup workGroup = (WorkGroup) groupDirectoryValues[ selectedIndex ];
909     URLString url = workGroup.getSessionURL();
910     Session session;
911     Channel channel;
912     String clientNames[] = null;
913     try
914     {
915         session = SessionFactory.createSession( getClientApplication().getClientManager().
916             getGroupManagementClient(), url, false );
917         channel = session.createChannel( getClientApplication().getClientManager().
918             getGroupManagementClient(),
919                 workGroup.getWorkGroupChannelID(),
920                 true, true, false);
921         clientNames = channel.listClientNames();
922     }
923     // the below is a remnant from an attempt to resolve the problem using a different method than a for
924     // loop cycling
925     // through the member names, comparing each to the unique name of this client.
926     catch ( NameInUseException NIUE )
927     {
928         MessageBox messageBox = new MessageBox();
929         messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
930         messageBox.show();
931         return;
932     }
933     catch ( JSDTEException JSDTE )
934     {
935         getClientApplication().getClientManager().handleJSDTEException( JSDTE );
936     }
937     for ( int i = 0 ; i < clientNames.length; i++ )
938     {
939         if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
940             getNAME() ) ) )
941         {
942             MessageBox messageBox = new MessageBox();
943             messageBox.setText( ClientConstants.getALREADY_JOINED_TO_GROUP_MESSAGE() );
944             messageBox.show();
945             return;
946         }
947     }
948     /* for ( int i = 0; i < clientNames.length; i++ )
949     {
950         // System.out.println( "Clientname: " + clientNames[i] );
951         // if client is already joined to the selected group
952         if ( clientNames[i].equalsIgnoreCase( ClientInfo.getPreferences().getProperty( ClientConstants.
953             getNAME() ) ) );
954     }

```

```

952     }
953     */
954     WorkGroup activeGroup = getClientApplication().getClientManager().getClientInfo().getActiveGroup();
955     getClientApplication().getClientManager().disconnectFromGroup( activeGroup );
956     getClientApplication().getClientManager().connectToGroup( workGroup );
957 }
958 /**
959  * main entrypoint - starts the part when it is run as an application
960  * @param args java.lang.String[]
961  */
962 public static void main(java.lang.String[] args) {
963     try {
964         GroupDirectory aGroupDirectory;
965         aGroupDirectory = new GroupDirectory();
966         aGroupDirectory.setModal(true);
967         aGroupDirectory.addWindowListener(new java.awt.event.WindowAdapter() {
968             public void windowClosing(java.awt.event.WindowEvent e) {
969                 System.exit(0);
970             }
971         });
972         aGroupDirectory.setVisible(true);
973     } catch (Throwable exception) {
974         System.err.println("Exception occurred in main() of javax.swing.JDialog");
975         exception.printStackTrace(System.out);
976     }
977 }
978 /**
979  * Insert the method's description here.
980  * Creation date: (10-08-00 15:41:17)
981  * @param newClientApplication clientapp.ClientApplication
982  */
983 public void setClientApplication(ClientApplication newClientApplication) {
984     clientApplication = newClientApplication;
985 }
986 /**
987  * Insert the method's description here.
988  * Creation date: (26-07-00 21:09:59)
989  * @param newGroupDirectoryValues peerviewmisc.WorkGroup[]
990  */
991 public void setGroupDirectoryValues(peerviewmisc.WorkGroup[] newGroupDirectoryValues) {
992     groupDirectoryValues = newGroupDirectoryValues;
993 }
994 /**
995  * Insert the method's description here.
996  * Creation date: (04-07-00 19:38:32)
997  * @param newTableModel javax.swing.table.DefaultTableModel
998  */
999 public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
1000     tableModel = newTableModel;
1001 }
1002 /**
1003  * Insert the method's description here.
1004  * Creation date: (03-07-00 21:00:14)
1005  * @param newWorkGroup peerviewmisc.WorkGroup
1006  */
1007 public void setWorkGroup(peerviewmisc.WorkGroup newWorkGroup) {
1008     workGroup = newWorkGroup;
1009 }
1010 /**
1011  * Update the table model underlying the group directory.
1012  * Creation date: (22-07-00 23:35:40)
1013  */
1014 public void updateTable()
1015 {
1016     if ( getClientApplication().getClientManager().getClientInfo().getGroups() == null )
1017     {

```

```

1018     disableButtons();
1019     getClientApplication().getClientManager().requestGroupsFromServer();
1020     return;
1021 }
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;
1029 while ( iterator.hasNext() )
{
1030     workGroup = (WorkGroup) iterator.next();
1031     Vector v = workGroup.getGroupAsVector();
1032     tableModel.addRow( v );
1033     groupDirectoryValues[i++] = workGroup;
1034 }
// Distribute surplus space evenly
1035 getScrollPaneTable().sizeColumnsToFit( -1 );
getScrollPaneTable().revalidate();
1036 getScrollPaneTable().repaint();
getJScrollPane1().revalidate();
1037 getJScrollPane1().repaint();

1038 }

```

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    {
17        public void actionPerformed( java.awt.event.ActionEvent ae )
18        {
19            dispose();
20        }
21    }
22    private javax.swing.JPanel ivjJDialogContentPane = null;
23    private javax.swing.JScrollPane ivjJScrollPane1 = null;
24    private javax.swing.JTable ivjScrollPaneTable = null;
25    private javax.swing.JButton ivjCloseButton = null;
26    private javax.swing.table.DefaultTableModel tableModel = new DefaultTableModel();
27    private javax.swing.JPanel ivjButtonPanel = null;
28    private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
29
30 /**
31 * MessagePropertiesBox constructor comment.
32 */
33 public MessagePropertiesBox()
34 {
    super();
    initialize();
}

```

```

36 /**
 * MessagePropertiesBox constructor comment.
38 * @param owner java.awt.Dialog
 */
40 public MessagePropertiesBox(java.awt.Dialog owner) {
    super(owner);
42 }
43 /**
44 * MessagePropertiesBox constructor comment.
45 * @param owner java.awt.Dialog
46 * @param title java.lang.String
 */
48 public MessagePropertiesBox(java.awt.Dialog owner, String title) {
    super(owner, title);
50 }
51 /**
52 * MessagePropertiesBox constructor comment.
53 * @param owner java.awt.Dialog
54 * @param title java.lang.String
55 * @param modal boolean
 */
56 public MessagePropertiesBox(java.awt.Dialog owner, String title, boolean modal) {
58     super(owner, title, modal);
}
59 /**
60 * MessagePropertiesBox constructor comment.
61 * @param owner java.awt.Dialog
62 * @param modal boolean
 */
63 public MessagePropertiesBox(java.awt.Dialog owner, boolean modal) {
65     super(owner, modal);
}
66 /**
67 * MessagePropertiesBox constructor comment.
68 * @param owner java.awt.Frame
 */
69 public MessagePropertiesBox(java.awt.Frame owner) {
    super(owner);
70 }
71 /**
72 * MessagePropertiesBox constructor comment.
73 * @param owner java.awt.Frame
74 * @param title java.lang.String
 */
75 public MessagePropertiesBox(java.awt.Frame owner, String title) {
    super(owner, title);
}
76 /**
77 * MessagePropertiesBox constructor comment.
78 * @param owner java.awt.Frame
79 * @param title java.lang.String
 */
80 public MessagePropertiesBox(java.awt.Frame owner, String title, boolean modal) {
82     super(owner, title, modal);
}
83 /**
84 * MessagePropertiesBox constructor comment.
85 * @param owner java.awt.Frame
86 * @param title java.lang.String
87 * @param modal boolean
 */
88 public MessagePropertiesBox(java.awt.Frame owner, String title, boolean modal) {
89     super(owner, title, modal);
}
90 /**
91 * MessagePropertiesBox constructor comment.
92 * @param owner java.awt.Frame
93 * @param modal boolean
 */
94 public MessagePropertiesBox(java.awt.Frame owner, boolean modal) {
95     super(owner, modal);
}
96 /**
97 * Return the JPanel1 property value.

```

```

102  * @return javax.swing.JPanel
103  */
104 /* WARNING: THIS METHOD WILL BE REGENERATED. */
105 private javax.swing.JPanel getButtonPanel() {
106     if (ivjButtonPanel == null) {
107         try {
108             ivjButtonPanel = new javax.swing.JPanel();
109             ivjButtonPanel.setName("ButtonPanel");
110             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
111             getButtonPanel().add(getCloseButton(), getName());
112             // user code begin {1}
113             // user code end
114         } catch (java.lang.Throwable ivjExc) {
115             // user code begin {2}
116             // user code end
117             handleException(ivjExc);
118         }
119     }
120     return ivjButtonPanel;
121 }
122 /**
123  * Return the ButtonPanelFlowLayout property value.
124  * @return java.awt.FlowLayout
125  */
126 /* WARNING: THIS METHOD WILL BE REGENERATED. */
127 private java.awt.FlowLayout getButtonPanelFlowLayout() {
128     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
129     try {
130         /* Create part */
131         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
132         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
133     } catch (java.lang.Throwable ivjExc) {
134         handleException(ivjExc);
135     };
136     return ivjButtonPanelFlowLayout;
137 }
138 /**
139  * Return the OKButton property value.
140  * @return javax.swing.JButton
141  */
142 /* WARNING: THIS METHOD WILL BE REGENERATED. */
143 private javax.swing.JButton getCloseButton() {
144     if (ivjCloseButton == null) {
145         try {
146             ivjCloseButton = new javax.swing.JButton();
147             ivjCloseButton.setName("CloseButton");
148             ivjCloseButton.setMnemonic('c');
149             ivjCloseButton.setText("Close");
150             // user code begin {1}
151             ivjCloseButton.addActionListener( new CloseButtonHandler() );
152             // user code end
153         } catch (java.lang.Throwable ivjExc) {
154             // user code begin {2}
155             // user code end
156             handleException(ivjExc);
157         }
158     }
159     return ivjCloseButton;
160 }
161 /**
162  * Return the JDialogContentPane property value.
163  * @return javax.swing.JPanel
164  */
165 /* WARNING: THIS METHOD WILL BE REGENERATED. */
166 private javax.swing.JPanel getJDialogContentPane() {
167     if (ivjJDialogContentPane == null) {

```

```

168     try {
169         ivjJDialogContentPane = new javax.swing.JPanel();
170         ivjJDialogContentPane.setName("JDialogContentPane");
171         ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
172         getJDialogContentPane().add(getButtonPanel(), "South");
173         getJDialogContentPane().add(getJScrollPane1(), "Center");
174         // user code begin {1}
175         // user code end
176     } catch (java.lang.Throwable ivjExc) {
177         // user code begin {2}
178         // user code end
179         handleException(ivjExc);
180     }
181 }
182 return ivjJDialogContentPane;
183 /**
184 * Return the JScrollPane1 property value.
185 * @return javax.swing.JScrollPane
186 */
187 /* WARNING: THIS METHOD WILL BE REGENERATED. */
188 private javax.swing.JScrollPane getJScrollPane1() {
189     if (ivjJScrollPane1 == null) {
190         try {
191             ivjJScrollPane1 = new javax.swing.JScrollPane();
192             ivjJScrollPane1.setName("JScrollPane1");
193             ivjJScrollPane1.setAutoscrolls(true);
194             ivjJScrollPane1.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
195             ivjJScrollPane1.setHorizontalScrollBarPolicy(javax.swing.JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
196                 );
197             getJScrollPane1().setViewportView(getScrollPaneTable());
198             // user code begin {1}
199             // user code end
200         } catch (java.lang.Throwable ivjExc) {
201             // user code begin {2}
202             // user code end
203             handleException(ivjExc);
204         }
205     }
206     return ivjJScrollPane1;
207 }
208 /**
209 * Return the ScrollPaneTable property value.
210 * @return javax.swing.JTable
211 */
212 /* WARNING: THIS METHOD WILL BE REGENERATED. */
213 private javax.swing.JTable getScrollPaneTable() {
214     if (ivjScrollPaneTable == null) {
215         try {
216             ivjScrollPaneTable = new javax.swing.JTable();
217             ivjScrollPaneTable.setName("ScrollPaneTable");
218             getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
219             getJScrollPane1().getViewport().setBackingStoreEnabled(true);
220             ivjScrollPaneTable.setBackground(java.awt.Color.white);
221             ivjScrollPaneTable.setForeground(java.awt.Color.black);
222             ivjScrollPaneTable.setGridColor(java.awt.SystemColor.desktop);
223             ivjScrollPaneTable.setFont(new java.awt.Font("dialog", 0, 12));
224             ivjScrollPaneTable.setIntercellSpacing(new java.awt.Dimension(5, 5));
225             ivjScrollPaneTable.setBounds(0, 0, 200, 200);
226             // user code begin {1}
227             ivjScrollPaneTable.setModel(TableModel);
228             ivjScrollPaneTable.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
229             ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
230
231             // user code end
232         } catch (java.lang.Throwable ivjExc) {

```

```

        // user code begin {2}
234     // user code end
235     handleException(ivjExc);
236 }
237 return ivjScrollPaneTable;
}
238 /**
239 * Insert the method's description here.
240 * Creation date: (19-07-00 19:46:21)
241 * @return javax.swing.table.DefaultTableModel
242 */
243 public javax.swing.table.DefaultTableModel getTableModel() {
244     return tableModel;
}
245 /**
246 * Called whenever the part throws an exception.
247 * @param exception java.lang.Throwable
248 */
249 private void handleException(java.lang.Throwable exception) {
250
251     /* Uncomment the following lines to print uncaught exceptions to stdout */
252     // System.out.println("----- UNCAUGHT EXCEPTION -----");
253     // exception.printStackTrace(System.out);
}
254 /**
255 * Initialize the class.
256 */
257 /* WARNING: THIS METHOD WILL BE REGENERATED. */
258 private void initialize() {
259     try {
260         // user code begin {1}
261         // user code end
262         setName("MessagePropertiesBox");
263         setDefaultCloseOperation(javax.WindowConstants.DISPOSE_ON_CLOSE);
264         setSize(415, 257);
265         setModal(true);
266         setTitle("MessageProperties");
267         setContentPane(getJDialogContentPane());
268     } catch (java.lang.Throwable ivjExc) {
269         handleException(ivjExc);
}
270 }
271 // user code begin {2}
272 // default positioning puts the dialog at center of screen. The method used is a slight hack, but
273 // works fine.
274 setLocationRelativeTo( getOwner() );
275 getScrollPaneTable().setAutoResizeMode( javax.swing.JTable.AUTO_RESIZE_OFF );
276 // user code end
}
277 /**
278 * Insert the method's description here.
279 * Creation date: (19-07-00 19:45:02)
280 * @param message peerviewmisc.Message
281 */
282 public void initTable(peerviewmisc.Message message)
{
283     // The below ought to be replaceable with calls to addColumn, but an attempt to do so didn't work as
284     // intended - perhaps a JDK bug ?
285     Object fieldNames[] = message.getFIELD_NAMES();
286     Vector values = message.getDataAsVector();
287     tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_NAME_COLUMN_HEADING() );
288     tableModel.addColumn( ClientConstants.getMESSAGE_PROPERTIES_VALUE_COLUMN_HEADING() );
289     for ( int i = 0; i < fieldNames.length; i++ )
{
290         Vector v = new Vector();
291         v.addElement( fieldNames[i] );
}

```

```

298     v.addElement( values.get(i) );
299     tableModel.addRow( v );
300 }
301 getScrollPaneTable().setColumnSelectionAllowed( false );
302 getScrollPaneTable().setRowSelectionAllowed( false );
303 getScrollPaneTable().setEnabled( false );
304 getScrollPaneTable().setCellEditor( null );
305 getScrollPaneTable().removeEditor();
306 /**
307  * main entrypoint - starts the part when it is run as an application
308  * @param args java.lang.String[]
309  */
310 public static void main(java.lang.String[] args) {
311     try {
312         MessagePropertiesBox aMessagePropertiesBox;
313         aMessagePropertiesBox = new MessagePropertiesBox();
314         aMessagePropertiesBox.setModal(true);
315         aMessagePropertiesBox.addWindowListener(new java.awt.event.WindowAdapter() {
316             public void windowClosing(java.awt.event.WindowEvent e) {
317                 System.exit(0);
318             }
319         });
320         aMessagePropertiesBox.setVisible(true);
321     } catch (Throwable exception) {
322         System.err.println("Exception occurred in main() of javax.swing.JDialog");
323         exception.printStackTrace(System.out);
324     }
325 /**
326  * Insert the method's description here.
327  * Creation date: (19-07-00 19:46:21)
328  * @param newTableModel javax.swing.table.DefaultTableModel
329  */
330     public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
331         tableModel = newTableModel;
332     }
333 }
334 }
```

Listing C.15: MessageWindow.java

```

package clientapp;
1
import java.awt.event.*;
2
import javax.swing.event.*;
3
4
6 /**
7  * This class implements the message window that can be activated by double clicking in the message bar
8  * at the bottom of the
9  * client application window.
10 * Creation date: (16-08-00 16:44:45)
11 * @author:
12 */
12 public class MessageWindow extends javax.swing.JDialog {
13
14     public class CloseItemHandler implements ActionListener
15     {
16         public void actionPerformed( ActionEvent ae )
17         {
18             closeMessageWindow();
19         }
20     }
21
22     public class SaveItemHandler implements ActionListener
23     {
24         public void actionPerformed( ActionEvent ae )
25         {
```

```

26         saveMessageWindowContents();
27     }
28 }
29 private javax.swing.JMenuItem ivjCloseItem = null;
30 private javax.swing.JMenu ivjFileMenu = null;
31 private javax.swing.JPanel ivjJDialogContentPane = null;
32 private javax.swing.JScrollPane ivjJScrollPane1 = null;
33 private javax.swing.JSeparator ivjJSeparator1 = null;
34 private javax.swing.JMenuBar ivjMessageWindowJMenuBar = null;
35 private javax.swing.JMenuItem ivjSaveItem = null;
36 private javax.swing.JTextArea ivjTextArea = null;
37     private ClientApplication clientApplication = null;
38 /**
39  * MessageWindow constructor comment.
40  */
41 public MessageWindow() {
42     super();
43     initialize();
44 }
45 /**
46  * MessageWindow constructor comment.
47  * @param owner java.awt.Dialog
48  */
49 public MessageWindow(java.awt.Dialog owner) {
50     super(owner);
51 }
52 /**
53  * MessageWindow constructor comment.
54  * @param owner java.awt.Dialog
55  * @param title java.lang.String
56  */
57 public MessageWindow(java.awt.Dialog owner, String title) {
58     super(owner, title);
59 }
60 /**
61  * MessageWindow constructor comment.
62  * @param owner java.awt.Dialog
63  * @param title java.lang.String
64  * @param modal boolean
65  */
66 public MessageWindow(java.awt.Dialog owner, String title, boolean modal) {
67     super(owner, title, modal);
68 }
69 /**
70  * MessageWindow constructor comment.
71  * @param owner java.awt.Dialog
72  * @param modal boolean
73  */
74 public MessageWindow(java.awt.Dialog owner, boolean modal) {
75     super(owner, modal);
76 }
77 /**
78  * MessageWindow constructor comment.
79  * @param owner java.awt.Frame
80  */
81 public MessageWindow(java.awt.Frame owner) {
82     super(owner);
83 }
84 /**
85  * MessageWindow constructor comment.
86  * @param owner java.awt.Frame
87  * @param title java.lang.String
88  */
89 public MessageWindow(java.awt.Frame owner, String title) {
90     super(owner, title);
91 }

```

```

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.
 * Creation date: (16-08-00 17:51:13)
 */
122 public void closeMessageWindow()
{
124     setVisible( false );
}
126 /**
 * Insert the method's description here.
 * Creation date: (29-10-00 17:08:48)
 * @return clientapp.ClientApplication
 */
128 public ClientApplication getClientApplication() {
130     return clientApplication;
}
132 }
134 /**
 * Return the CloseItem property value.
 * @return javax.swing.JMenuItem
 */
136 /* WARNING: THIS METHOD WILL BE REGENERATED. */
138 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.
159 * @return javax.swing.JMenu
160 */
161 /* WARNING: THIS METHOD WILL BE REGENERATED. */
162 private javax.swing.JMenu getFileMenu() {
163     if (ivjFileMenu == null) {
164         try {
165             ivjFileMenu = new javax.swing.JMenu();
166             ivjFileMenu.setName("FileMenu");
167             ivjFileMenu.setMnemonic('F');
168             ivjFileMenu.setText("File");
169             ivjFileMenu.add(getSaveItem());
170             ivjFileMenu.add(getJSeparator1());
171             ivjFileMenu.add(getCloseItem());
172             // user code begin {1}
173             // user code end
174         } catch (java.lang.Throwable ivjExc) {
175             // user code begin {2}
176             // user code end
177             handleException(ivjExc);
178         }
179     }
180     return ivjFileMenu;
181 }
182 /**
183 * Return the JDialgContentPane property value.
184 * @return javax.swing.JPanel
185 */
186 /* WARNING: THIS METHOD WILL BE REGENERATED. */
187 private javax.swing.JPanel getJDialgContentPane() {
188     if (ivjJDialgContentPane == null) {
189         try {
190             ivjJDialgContentPane = new javax.swing.JPanel();
191             ivjJDialgContentPane.setName("JDialgContentPane");
192             ivjJDialgContentPane.setLayout(new java.awt.BorderLayout());
193             getJDialgContentPane().add(getJScrollPane1(), "Center");
194             // user code begin {1}
195             // user code end
196         } catch (java.lang.Throwable ivjExc) {
197             // user code begin {2}
198             // user code end
199             handleException(ivjExc);
200         }
201     }
202     return ivjJDialgContentPane;
203 }
204 /**
205 * Return the JScrollPane1 property value.
206 * @return javax.swing.JScrollPane
207 */
208 /* WARNING: THIS METHOD WILL BE REGENERATED. */
209 private javax.swing.JScrollPane getJScrollPane1() {
210     if (ivjJScrollPane1 == null) {
211         try {
212             ivjJScrollPane1 = new javax.swing.JScrollPane();
213             ivjJScrollPane1.setName("JScrollPane1");
214             getJScrollPane1().setViewportView(getTextArea());
215             // user code begin {1}
216             // user code end
217         } catch (java.lang.Throwable ivjExc) {
218             // user code begin {2}
219             // user code end
220             handleException(ivjExc);
221         }
222     }
223     return ivjJScrollPane1;

```

```

224 }
225 /**
226 * Return the JSeparator1 property value.
227 * @return javax.swing.JSeparator
228 */
229 /* WARNING: THIS METHOD WILL BE REGENERATED. */
230 private javax.swing.JSeparator getJSeparator1() {
231     if (ivjJSeparator1 == null) {
232         try {
233             ivjJSeparator1 = new javax.swing.JSeparator();
234             ivjJSeparator1.setName("JSeparator1");
235             // user code begin {1}
236             // user code end
237         } catch (java.lang.Throwable ivjExc) {
238             // user code begin {2}
239             // user code end
240             handleException(ivjExc);
241         }
242     }
243     return ivjJSeparator1;
244 }
245 /**
246 * Return the MessageWindowJMenuBar property value.
247 * @return javax.swing.JMenuBar
248 */
249 /* WARNING: THIS METHOD WILL BE REGENERATED. */
250 private javax.swing.JMenuBar getMessageWindowJMenuBar() {
251     if (ivjMessageWindowJMenuBar == null) {
252         try {
253             ivjMessageWindowJMenuBar = new javax.swing.JMenuBar();
254             ivjMessageWindowJMenuBar.setName("MessageWindowJMenuBar");
255             ivjMessageWindowJMenuBar.add(getFileMenu());
256             // user code begin {1}
257             // user code end
258         } catch (java.lang.Throwable ivjExc) {
259             // user code begin {2}
260             // user code end
261             handleException(ivjExc);
262         }
263     }
264     return ivjMessageWindowJMenuBar;
265 }
266 /**
267 * Return the SaveItem property value.
268 * @return javax.swing.JMenuItem
269 */
270 /* WARNING: THIS METHOD WILL BE REGENERATED. */
271 private javax.swing.JMenuItem getSaveItem() {
272     if (ivjSaveItem == null) {
273         try {
274             ivjSaveItem = new javax.swing.JMenuItem();
275             ivjSaveItem.setName("SaveItem");
276             ivjSaveItem.setMnemonic('S');
277             ivjSaveItem.setText("Save");
278             // user code begin {1}
279             ivjSaveItem.addActionListener( new SaveItemHandler() );
280             // user code end
281         } catch (java.lang.Throwable ivjExc) {
282             // user code begin {2}
283             // user code end
284             handleException(ivjExc);
285         }
286     }
287     return ivjSaveItem;
288 }
289 /**

```

```

290 * Return the TextArea property value.
291 * @return javax.swing.JTextPane
292 */
293 /* WARNING: THIS METHOD WILL BE REGENERATED. */
294 private javax.swing.JTextArea getTextArea() {
295     if (ivjTextArea == null) {
296         try {
297             ivjTextArea = new javax.swing.JTextArea();
298             ivjTextArea.setName("TextArea");
299             ivjTextArea.setToolTipText("Messages generated by the peer view client application.");
300             ivjTextArea.setBounds(0, 0, 7, 6);
301             // user code begin {1}
302             // user code end
303         } catch (java.lang.Throwable ivjExc) {
304             // user code begin {2}
305             // user code end
306             handleException(ivjExc);
307         }
308     }
309     return ivjTextArea;
310 }
311 /**
312 * Called whenever the part throws an exception.
313 * @param exception java.lang.Throwable
314 */
315 private void handleException(java.lang.Throwable exception) {
316     /* Uncomment the following lines to print uncaught exceptions to stdout */
317     // System.out.println("----- UNCAUGHT EXCEPTION -----");
318     // exception.printStackTrace(System.out);
319 }
320 /**
321 * Initialize the class.
322 */
323 /* WARNING: THIS METHOD WILL BE REGENERATED. */
324 private void initialize() {
325     try {
326         // user code begin {1}
327         // user code end
328         setName("MessageWindow");
329         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
330         setTitle("MessageWindow");
331         setSize(512, 479);
332         setVisible(false);
333         setJMenuBar(getMessageWindowJMenuBar());
334         setContentPane(getJDialogContentPane());
335     } catch (java.lang.Throwable ivjExc) {
336         handleException(ivjExc);
337     }
338     // user code begin {2}
339     getTextArea().setEditable( false );
340     getTextArea().setLineWrap( false );
341     setDefaultCloseOperation( javax.swing.JDialog.HIDE_ON_CLOSE );
342     setLocationRelativeTo( getOwner() );
343     // user code end
344 }
345 /**
346 * main entrypoint - starts the part when it is run as an application
347 * @param args java.lang.String[]
348 */
349 public static void main(java.lang.String[] args) {
350     try {
351         MessageWindow aMessageWindow;
352         aMessageWindow = new MessageWindow();
353         aMessageWindow.setModal(true);
354         aMessageWindow.addWindowListener(new java.awt.event.WindowAdapter() {

```

```

356     public void windowClosing(java.awt.event.WindowEvent e) {
357         System.exit(0);
358     };
359     aMessageWindow.setVisible(true);
360 } catch (Throwable exception) {
361     System.err.println("Exception occurred in main() of javax.swing.JDialog");
362     exception.printStackTrace(System.out);
363 }
364 /**
365 * Save the contents of the message window to disk.
366 * Creation date: (29-10-00 16:34:58)
367 */
370 public void saveMessageWindowContents()
371 {
372     try
373     {
374         java.io.FileOutputStream fos = new java.io.FileOutputStream( ClientInfo.getPreferences().
375             getProperty( ClientConstants.getMESSAGE_LOG_FILENAME() ) );
376         java.io.ObjectOutputStream oos = new java.io.ObjectOutputStream( fos );
377
378         oos.writeUTF( getTextArea().getText() );
379         oos.flush();
380         oos.close();
381     }
382     catch ( Exception E )
383     {
384         System.err.println( ClientConstants.getMESSAGE_LOG_WRITE_ERROR() );
385     }
386 }
387 /**
388 * Insert the method's description here.
389 * Creation date: (29-10-00 17:08:48)
390 * @param newClientApplication clientapp.ClientApplication
391 */
392 public void setClientApplication(ClientApplication newClientApplication) {
393     clientApplication = newClientApplication;
394 }
395 }
```

Listing C.16: PanoramaLayout.java

```

package clientapp;
2
3 /**
4  * Abstract superclass for client panorama layout managers.
5  * Creation date: (22-06-00 11:30:49)
6  * @author:
7  */
8 import java.awt.Dimension;

10 public abstract class PanoramaLayout
11 {
12     protected java.awt.Dimension preferredDocumentSize = new Dimension();
13     protected int xSpacing;
14     protected int ySpacing;
15     protected java.awt.Dimension previousPlacement = null;
16     protected int horizontalIndex = 0;
17     protected int verticalIndex = 0;
18 /**
19  * PanoramaLayout constructor comment.
20 */
21     public PanoramaLayout( )
22 {
```

```

preferredDocumentSize = new Dimension( Integer.parseInt( (String) ClientInfo.getPreferences().
24                               getProperty(
25                                 ClientConstants.DEFAULT_DOCUMENT_WIDTH() ) ),
26                               Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
27                                 ClientConstants.DEFAULT_DOCUMENT_HEIGHT() ) ) );
28 xSpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
29                                 ClientConstants.DEFAULT_HORIZONTAL_DOCUMENT_SPACING() ) );
30 ySpacing = Integer.parseInt( (String) ClientInfo.getPreferences().getProperty(
31                                 ClientConstants.DEFAULT_VERTICAL_DOCUMENT_SPACING() ) );
32 }
32 /**
33 * Insert the method's description here.
34 * Creation date: (22-06-00 11:50:28)
35 * @return java.awt.Dimension
36 */
37 public abstract Dimension computePlacement() throws Exception;
38 /**
39 * Insert the method's description here.
40 * Creation date: (01-08-00 15:42:44)
41 * @return int
42 */
43 public int getHorizontalIndex() {
44     return horizontalIndex;
45 }
46 /**
47 * Insert the method's description here.
48 * Creation date: (01-08-00 19:42:20)
49 * @return java.awt.Rectangle
50 */
51 public abstract java.awt.Rectangle getOverviewRectangle();
52 /**
53 * Insert the method's description here.
54 * Creation date: (22-06-00 11:32:36)
55 * @return java.awt.Dimension
56 */
57 public java.awt.Dimension getPreferredDocumentSize() {
58     return preferredDocumentSize;
59 }
60 /**
61 * Insert the method's description here.
62 * Creation date: (22-06-00 12:02:54)
63 * @return java.awt.Dimension
64 */
65 public java.awt.Dimension getPreviousPlacement() {
66     return previousPlacement;
67 }
68 /**
69 * Insert the method's description here.
70 * Creation date: (01-08-00 15:42:53)
71 * @return int
72 */
73 public int getVerticalIndex() {
74     return verticalIndex;
75 }
76 /**
77 * Insert the method's description here.
78 * Creation date: (22-06-00 11:37:05)
79 * @return int
80 */
81 public int getXSpacing() {
82     return xSpacing;
83 }
84 /**
85 * Insert the method's description here.
86 * Creation date: (22-06-00 11:37:25)
87 * @return int

```

```

88     */
89     public int getYSpacing() {
90         return ySpacing;
91     }
92     /**
93      * Insert the method's description here.
94      * Creation date: (23-06-00 22:22:43)
95      */
96     public abstract void resetLayout();
97     /**
98      * Insert the method's description here.
99      * Creation date: (01-08-00 15:42:44)
100     * @param newHorizontalIndex int
101     */
102    public void setHorizontalIndex(int newHorizontalIndex) {
103        horizontalIndex = newHorizontalIndex;
104    }
105    /**
106     * Insert the method's description here.
107     * Creation date: (22-06-00 11:32:36)
108     * @param newPreferredDocumentSize java.awt.Dimension
109     */
110    public void setPreferredDocumentSize(java.awt.Dimension newPreferredDocumentSize) {
111        preferredDocumentSize = newPreferredDocumentSize;
112    }
113    /**
114     * Insert the method's description here.
115     * Creation date: (22-06-00 12:02:54)
116     * @param newPreviousPlacement java.awt.Dimension
117     */
118    public void setPreviousPlacement(java.awt.Dimension newPreviousPlacement) {
119        previousPlacement = newPreviousPlacement;
120    }
121    /**
122     * Insert the method's description here.
123     * Creation date: (01-08-00 15:42:53)
124     * @param newVerticalIndex int
125     */
126    public void setVerticalIndex(int newVerticalIndex) {
127        verticalIndex = newVerticalIndex;
128    }
129    /**
130     * Insert the method's description here.
131     * Creation date: (22-06-00 11:37:05)
132     * @param newXSpacing int
133     */
134    public void setXSpacing(int newXSpacing) {
135        xSpacing = newXSpacing;
136    }
137    /**
138     * Insert the method's description here.
139     * Creation date: (22-06-00 11:37:25)
140     * @param newYSpacing int
141     */
142    public void setYSpacing(int newYSpacing) {
143        ySpacing = newYSpacing;
144    }
145 }
```

Listing C.17: PeerViewClient.java

```

package clientapp;
2
3 import com.sun.media.jsdt.*;
4 import com.sun.media.jsdt.event.*;
5 /**
6  * Insert the type's description here.
```

```

  * Creation date: (29-06-00 20:26:21)
8   * @author:
9   */
10  public class PeerViewClient implements Client, ChannelConsumer {
11      private java.lang.String name;
12  /**
13      * PeerViewClient constructor comment.
14  */
15  public PeerViewClient(String nameArg)
16  {
17      super();
18      name = nameArg;
19  }
20 /**
21     * authenticate method comment.
22 */
23  public Object authenticate(AuthenticationInfo arg1) {
24      return null;
25  }
26 /**
27     * Insert the method's description here.
28     * Creation date: (30-06-00 09:42:34)
29     * @param date com.sun.media.jsdt.Data
30     */
31  public synchronized void dataReceived(Data data)
32  {
33      // System.out.println( data.getDataAsString() );
34  }
35 /**
36     * Insert the method's description here.
37     * Creation date: (30-06-00 09:00:06)
38     * @return java.lang.String
39     */
40  public java.lang.String getName()
41  {
42      return name;
43  }
44 /**
45     * Insert the method's description here.
46     * Creation date: (30-06-00 09:00:06)
47     * @param newName java.lang.String
48     */
49  public void setName(java.lang.String newName) {
50      name = newName;
51  }
52 }

```

Listing C.18: PreferencesDialog.java

```

package clientapp;
2
3 import javax.swing.*;
4 import javax.swing.border.*;
5 import javax.swing.event.*;
6 import java.util.Properties;
7 import java.io.*;
8 import java.awt.event.*;
9 import java.util.Hashtable;
10 import java.util.HashSet;
11 import java.util.*;
12 import peerviewmisc.*;

14 /**
15     * This class implements the preferences dialog and its associated panes.
16 */
17 public class PreferencesDialog extends JDialog
18 {
19     public class Changer implements ChangeListener

```

```

20     {
21         public void stateChanged( ChangeEvent ce )
22         {
23             // System.out.println( getAnimationSpeedSlider().getValue() );
24         }
25     };
26
27     public class OKHandler implements ActionListener
28     {
29         public void actionPerformed (ActionEvent ae)
30         {
31             submitAndClose();
32         }
33     }
34
35     public class CancelHandler implements ActionListener
36     {
37         public void actionPerformed (ActionEvent ae)
38         {
39             cancelAndClose();
40         }
41     }
42
43     public class ConnectHandler implements ActionListener
44     {
45         public void actionPerformed (ActionEvent ae)
46         {
47             connectToServer();
48         }
49     }
50
51     public class CustomizeHandler implements ActionListener
52     {
53         public void actionPerformed( ActionEvent ae )
54         {
55             customizeLayout();
56         }
57     };
58
59     private JPanel ivjJDialogContentPane = null;
60     private JTabbedPane ivjJTabbedPane1 = null;
61     private JPanel ivjDisplayPage = null;
62     private JLabel ivjNameLabel = null;
63     private JPanel ivjPersonalPage = null;
64     private JLabel ivjSignatureLabel = null;
65     private JButton ivjCancelButton = null;
66     private JPanel ivj JPanel1 = null;
67     private java.awt.FlowLayout ivj JPanel1FlowLayout = null;
68     private JButton ivjOKButton = null;
69     private JScrollPane ivj JScrollPane1 = null;
70     private ClientApplication clientApplication = null;
71     private JButton ivjCustomizeButton = null;
72     private JComboBox ivjDisplayQualityBox = null;
73     private JLabel ivjDisplayQualityLabel = null;
74     private JComboBox ivjLayoutSchemeBox = null;
75     private JLabel ivjLayoutSchemeLabel = null;
76     IvjEventHandler ivjEventHandler = new IvjEventHandler();
77     private JTextField ivjNameTextField = null;
78     private JTextPane ivjSignaturePane = null;
79     private JLabel ivjServerName = null;
80     private JPanel ivjServerPage = null;
81     private JLabel ivjServerPortLabel = null;
82     private JTextField ivjServerNameField = null;
83     private JTextField ivjServerPortField = null;
84     private JLabel ivjConnectionType = null;
85     private JComboBox ivjConnectionTypeBox = null;

```

```

    private JTextField ivjUpdateIntervalField = null;
86  private JLabel ivjUpdateIntervalLabel = null;
    private JButton ivjConnectButton = null;
88  private JLabel ivjAnimationSpeedLabel = null;
    private JLabel ivjFastLabel = null;
90  private JLabel ivjSlowLabel = null;
    private JSlider ivjAnimationSpeedSlider = null;
92
93  class IvjEventHandler implements java.awt.event.ActionListener {
94      public void actionPerformed(java.awt.event.ActionEvent e) {
95          if (e.getSource() == PreferencesDialog.this.getCancelButton())
96              connEtoM1(e);
97      }
98  };
99  /**
100   * PreferencesDialog constructor comment.
101   */
102  public PreferencesDialog() {
103      super();
104      initialize();
105  }
106 /**
107  * PreferencesDialog constructor comment.
108  * @param owner java.awt.Dialog
109  */
110  public PreferencesDialog(java.awt.Dialog owner) {
111      super(owner);
112  }
113 /**
114  * PreferencesDialog constructor comment.
115  * @param owner java.awt.Dialog
116  * @param title java.lang.String
117  */
118  public PreferencesDialog(java.awt.Dialog owner, String title) {
119      super(owner, title);
120  }
121 /**
122  * PreferencesDialog constructor comment.
123  * @param owner java.awt.Dialog
124  * @param title java.lang.String
125  * @param modal boolean
126  */
127  public PreferencesDialog(java.awt.Dialog owner, String title, boolean modal) {
128      super(owner, title, modal);
129  }
130 /**
131  * PreferencesDialog constructor comment.
132  * @param owner java.awt.Dialog
133  * @param modal boolean
134  */
135  public PreferencesDialog(java.awt.Dialog owner, boolean modal) {
136      super(owner, modal);
137  }
138 /**
139  * PreferencesDialog constructor comment.
140  * @param owner java.awt.Frame
141  */
142  public PreferencesDialog(java.awt.Frame owner) {
143      super(owner);
144  }
145 /**
146  * PreferencesDialog constructor comment.
147  * @param owner java.awt.Frame
148  * @param title java.lang.String
149  */
150  public PreferencesDialog(java.awt.Frame owner, String title) {

```

```

        super(owner, title);
152    }
153    /**
154     * PreferencesDialog constructor comment.
155     * @param owner java.awt.Frame
156     * @param title java.lang.String
157     * @param modal boolean
158     */
159    public PreferencesDialog(java.awt.Frame owner, String title, boolean modal) {
160        super(owner, title, modal);
161    }
162    /**
163     * PreferencesDialog constructor comment.
164     * @param owner java.awt.Frame
165     * @param modal boolean
166     */
167    public PreferencesDialog(java.awt.Frame owner, boolean modal) {
168        super(owner, modal);
169    }
170    /**
171     * Insert the method's description here.
172     * Creation date: (05-08-00 20:27:21)
173     */
174    public void cancelAndClose()
175    {
176        dispose();
177    }
178    /**
179     * Insert the method's description here.
180     * Creation date: (10-08-00 20:27:16)
181     */
182    public void connectToServer()
183    {
184        ConfirmationBox confirmationBox = new ConfirmationBox();
185        confirmationBox.setText( ClientConstants.getCONFIRM_CONNECT_TO_SERVER() );
186        confirmationBox.show();
187        if ( confirmationBox.isConfirmed() )
188        {
189            getClientApplication().getClientManager().connectToGroupManagement();
190        }
191    }
192    /**
193     * connEtoM1: (CancelButton.action.actionPerformed(java.awt.event.ActionEvent) --> PreferencesDialog.
194     *             dispose()V)
195     * @param arg1 java.awt.event.ActionEvent
196     */
197    /* WARNING: THIS METHOD WILL BE REGENERATED. */
198    private void connEtoM1(java.awt.event.ActionEvent arg1) {
199        try {
200            // user code begin {1}
201            // user code end
202            this.dispose();
203            // user code begin {2}
204            // user code end
205        } catch (java.lang.Throwable ivjExc) {
206            // user code begin {3}
207            // user code end
208            handleException(ivjExc);
209        }
210    }
211    /**
212     * Insert the method's description here.
213     * Creation date: (06-09-00 02:13:19)
214     */
215    public void customizeLayout()
216    {

```

```

216     getClientApplication().getPanoramaPanel().getLayoutManager().show( this );
217     getClientApplication().getPanoramaPanel().getLayoutManager().updateSettings( ClientInfo.
218         getPreferences() );
219 }
220 /**
221 * Return the AnimationSpeedLabel property value.
222 * @return javax.swing.JLabel
223 */
224 /* WARNING: THIS METHOD WILL BE REGENERATED. */
225 private javax.swing.JLabel getAnimationSpeedLabel() {
226     if (ivjAnimationSpeedLabel == null) {
227         try {
228             ivjAnimationSpeedLabel = new javax.swing.JLabel();
229             ivjAnimationSpeedLabel.setName("AnimationSpeedLabel");
230             ivjAnimationSpeedLabel.setText("AnimationSpeed:");
231             // user code begin {1}
232             // user code end
233         } catch (java.lang.Throwable ivjExc) {
234             // user code begin {2}
235             // user code end
236             handleException(ivjExc);
237         }
238     }
239     return ivjAnimationSpeedLabel;
240 }
241 /**
242 * Return the JSlider1 property value.
243 * @return javax.swing.JSlider
244 */
245 /* WARNING: THIS METHOD WILL BE REGENERATED. */
246 private javax.swing.JSlider getAnimationSpeedSlider() {
247     if (ivjAnimationSpeedSlider == null) {
248         try {
249             ivjAnimationSpeedSlider = new javax.swing.JSlider();
250             ivjAnimationSpeedSlider.setName("AnimationSpeedSlider");
251             // user code begin {1}
252             ivjAnimationSpeedSlider.setMinimum( ClientConstants.getANIMATION_MIN_SPEED() );
253             ivjAnimationSpeedSlider.setMaximum( ClientConstants.getANIMATION_MAX_SPEED() );
254             ivjAnimationSpeedSlider.setPaintTicks( true );
255             ivjAnimationSpeedSlider.setPaintTrack( true );
256             ivjAnimationSpeedSlider.addChangeListener( new Changer() );
257             // user code end
258         } catch (java.lang.Throwable ivjExc) {
259             // user code begin {2}
260             // user code end
261             handleException(ivjExc);
262         }
263     }
264     return ivjAnimationSpeedSlider;
265 }
266 /**
267 * Return the CancelButton property value.
268 * @return javax.swing.JButton
269 */
270 /* WARNING: THIS METHOD WILL BE REGENERATED. */
271 private javax.swing.JButton getCancelButton() {
272     if (ivjCancelButton == null) {
273         try {
274             ivjCancelButton = new javax.swing.JButton();
275             ivjCancelButton.setName("CancelButton");
276             ivjCancelButton.setText("Cancel");
277             // user code begin {1}
278             ivjCancelButton.addActionListener( new CancelHandler() );
279             // user code end
280         } catch (java.lang.Throwable ivjExc) {
281             // user code begin {2}

```

```

282         // user code end
283         handleException(ivjExc);
284     }
285     return ivjCancelButton;
286 }
287 /**
288 * Insert the method's description here.
289 * Creation date: (25-06-00 20:14:31)
290 * @return clientapp.ClientApplication
291 */
292 public ClientApplication getClientApplication() {
293     return clientApplication;
294 }
295 /**
296 * Return the ConnectButton property value.
297 * @return javax.swing.JButton
298 */
299 /* WARNING: THIS METHOD WILL BE REGENERATED. */
300 private javax.swing.JButton getConnectButton() {
301     if (ivjConnectButton == null) {
302         try {
303             ivjConnectButton = new javax.swing.JButton();
304             ivjConnectButton.setName("ConnectButton");
305             ivjConnectButton.setMnemonic('c');
306             ivjConnectButton.setText("Connect_to_server");
307             // user code begin {1}
308             ivjConnectButton.setEnabled( false );
309             ivjConnectButton.addActionListener( new ConnectHandler() );
310             // user code end
311         } catch (java.lang.Throwable ivjExc) {
312             // user code begin {2}
313             // user code end
314             handleException(ivjExc);
315         }
316     }
317     return ivjConnectButton;
318 }
319 /**
320 * Return the ConnectionType property value.
321 * @return javax.swing.JLabel
322 */
323 /* WARNING: THIS METHOD WILL BE REGENERATED. */
324 private javax.swing.JLabel getConnectionType() {
325     if (ivjConnectionType == null) {
326         try {
327             ivjConnectionType = new javax.swing.JLabel();
328             ivjConnectionType.setName("ConnectionType");
329             ivjConnectionType.setText("Connection_type:");
330             // user code begin {1}
331             // user code end
332         } catch (java.lang.Throwable ivjExc) {
333             // user code begin {2}
334             // user code end
335             handleException(ivjExc);
336         }
337     }
338     return ivjConnectionType;
339 }
340 /**
341 * Return the ConnectionTypeBox property value.
342 * @return javax.swing.JComboBox
343 */
344 /* WARNING: THIS METHOD WILL BE REGENERATED. */
345 private javax.swing.JComboBox getConnectionTypeBox() {
346     if (ivjConnectionTypeBox == null) {

```

```

try {
348     ivjConnectionTypeBox = new javax.swing.JComboBox();
349     ivjConnectionTypeBox.setName("ConnectionTypeBox");
350     // user code begin {1}
351     // user code end
352 } catch (java.lang.Throwable ivjExc) {
353     // user code begin {2}
354     // user code end
355     handleException(ivjExc);
356 }
357
358 return ivjConnectionTypeBox;
}
359 /**
360 * Return the CustomizeButton property value.
361 * @return javax.swing.JButton
362 */
363 /* WARNING: THIS METHOD WILL BE REGENERATED. */
364 private javax.swing.JButton getCustomizeButton() {
365     if (ivjCustomizeButton == null) {
366         try {
367             ivjCustomizeButton = new javax.swing.JButton();
368             ivjCustomizeButton.setName("CustomizeButton");
369             ivjCustomizeButton.setMnemonic('u');
370             ivjCustomizeButton.setText("Customize");
371             // user code begin {1}
372             ivjCustomizeButton.addActionListener( new CustomizeHandler() );
373             // user code end
374         } catch (java.lang.Throwable ivjExc) {
375             // user code begin {2}
376             // user code end
377             handleException(ivjExc);
378         }
379     }
380     return ivjCustomizeButton;
}
381 /**
382 * Return the DisplayPage property value.
383 * @return javax.swing.JPanel
384 */
385 /* WARNING: THIS METHOD WILL BE REGENERATED. */
386 private javax.swing.JPanel getDisplayPage() {
387     if (ivjDisplayPage == null) {
388         try {
389             ivjDisplayPage = new javax.swing.JPanel();
390             ivjDisplayPage.setName("DisplayPage");
391             ivjDisplayPage.setLayout(new java.awt.GridBagLayout());
392
393             java.awt.GridBagConstraints constraintsDisplayQualityLabel = new java.awt.GridBagConstraints();
394             constraintsDisplayQualityLabel.gridx = 0; constraintsDisplayQualityLabel.gridy = 0;
395             constraintsDisplayQualityLabel.gridwidth = 2;
396             constraintsDisplayQualityLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
397             constraintsDisplayQualityLabel.weighty = 1.0;
398             constraintsDisplayQualityLabel.insets = new java.awt.Insets(6, 6, 6, 6);
399             getDisplayPage().add(getDisplayQualityLabel(), constraintsDisplayQualityLabel);
400
401             java.awt.GridBagConstraints constraintsLayoutSchemeLabel = new java.awt.GridBagConstraints();
402             constraintsLayoutSchemeLabel.gridx = 0; constraintsLayoutSchemeLabel.gridy = 1;
403             constraintsLayoutSchemeLabel.gridwidth = 2;
404             constraintsLayoutSchemeLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
405             constraintsLayoutSchemeLabel.weighty = 1.0;
406             constraintsLayoutSchemeLabel.insets = new java.awt.Insets(6, 6, 6, 6);
407             getDisplayPage().add(getLayoutSchemeLabel(), constraintsLayoutSchemeLabel);
408
409             java.awt.GridBagConstraints constraintsDisplayQualityBox = new java.awt.GridBagConstraints();
410             constraintsDisplayQualityBox.gridx = 2; constraintsDisplayQualityBox.gridy = 0;
411
412         }
413     }
414     return ivjDisplayPage;
}

```

```

414     constraintsDisplayQualityBox.gridwidth = 2;
415     constraintsDisplayQualityBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
416     constraintsDisplayQualityBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
417     constraintsDisplayQualityBox.weightx = 0.3;
418     constraintsDisplayQualityBox.weighty = 1.0;
419     constraintsDisplayQualityBox.insets = new java.awt.Insets(6, 6, 6, 6);
420     getDisplayPage().add(getDisplayQualityBox(), constraintsDisplayQualityBox);
421
422     java.awt.GridBagConstraints constraintsLayoutSchemeBox = new java.awt.GridBagConstraints();
423     constraintsLayoutSchemeBox.gridx = 2; constraintsLayoutSchemeBox.gridy = 1;
424     constraintsLayoutSchemeBox.fill = java.awt.GridBagConstraints.HORIZONTAL;
425     constraintsLayoutSchemeBox.anchor = java.awt.GridBagConstraints.NORTHEAST;
426     constraintsLayoutSchemeBox.weightx = 0.3;
427     constraintsLayoutSchemeBox.weighty = 1.0;
428     constraintsLayoutSchemeBox.insets = new java.awt.Insets(6, 6, 6, 6);
429     getDisplayPage().add(getLayoutSchemeBox(), constraintsLayoutSchemeBox);
430
431     java.awt.GridBagConstraints constraintsCustomizeButton = new java.awt.GridBagConstraints();
432     constraintsCustomizeButton.gridx = 3; constraintsCustomizeButton.gridy = 1;
433     constraintsCustomizeButton.anchor = java.awt.GridBagConstraints.NORTHEAST;
434     constraintsCustomizeButton.insets = new java.awt.Insets(4, 4, 4, 4);
435     getDisplayPage().add(getCustomizeButton(), constraintsCustomizeButton);
436
437     java.awt.GridBagConstraints constraintsUpdateIntervalLabel = new java.awt.GridBagConstraints();
438     constraintsUpdateIntervalLabel.gridx = 1; constraintsUpdateIntervalLabel.gridy = 2;
439     constraintsUpdateIntervalLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
440     constraintsUpdateIntervalLabel.weighty = 1.0;
441     constraintsUpdateIntervalLabel.insets = new java.awt.Insets(4, 4, 4, 4);
442     getDisplayPage().add(getUpdateIntervalLabel(), constraintsUpdateIntervalLabel);
443
444     java.awt.GridBagConstraints constraintsUpdateIntervalField = new java.awt.GridBagConstraints();
445     constraintsUpdateIntervalField.gridx = 2; constraintsUpdateIntervalField.gridy = 2;
446     constraintsUpdateIntervalField.anchor = java.awt.GridBagConstraints.NORTHWEST;
447     constraintsUpdateIntervalField.weightx = 1.0;
448     constraintsUpdateIntervalField.weighty = 1.0;
449     constraintsUpdateIntervalField.ipadx = 100;
450     constraintsUpdateIntervalField.insets = new java.awt.Insets(4, 4, 4, 4);
451     getDisplayPage().add(getUpdateIntervalField(), constraintsUpdateIntervalField);
452
453     java.awt.GridBagConstraints constraintsAnimationSpeedLabel = new java.awt.GridBagConstraints();
454     constraintsAnimationSpeedLabel.gridx = 1; constraintsAnimationSpeedLabel.gridy = 4;
455     constraintsAnimationSpeedLabel.anchor = java.awt.GridBagConstraints.WEST;
456     constraintsAnimationSpeedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
457     getDisplayPage().add(getAnimationSpeedLabel(), constraintsAnimationSpeedLabel);
458
459     java.awt.GridBagConstraints constraintsAnimationSpeedSlider = new java.awt.GridBagConstraints();
460     constraintsAnimationSpeedSlider.gridx = 2; constraintsAnimationSpeedSlider.gridy = 4;
461     constraintsAnimationSpeedSlider.gridwidth = 2;
462     constraintsAnimationSpeedSlider.fill = java.awt.GridBagConstraints.HORIZONTAL;
463     constraintsAnimationSpeedSlider.anchor = java.awt.GridBagConstraints.EAST;
464     constraintsAnimationSpeedSlider.weightx = 1.0;
465     constraintsAnimationSpeedSlider.insets = new java.awt.Insets(4, 4, 4, 4);
466     getDisplayPage().add(getAnimationSpeedSlider(), constraintsAnimationSpeedSlider);
467
468     java.awt.GridBagConstraints constraintsSlowLabel = new java.awt.GridBagConstraints();
469     constraintsSlowLabel.gridx = 2; constraintsSlowLabel.gridy = 3;
470     constraintsSlowLabel.anchor = java.awt.GridBagConstraints.WEST;
471     constraintsSlowLabel.insets = new java.awt.Insets(4, 4, 4, 4);
472     getDisplayPage().add(getSlowLabel(), constraintsSlowLabel);
473
474     java.awt.GridBagConstraints constraintsFastLabel = new java.awt.GridBagConstraints();
475     constraintsFastLabel.gridx = 3; constraintsFastLabel.gridy = 3;
476     constraintsFastLabel.anchor = java.awt.GridBagConstraints.EAST;
477     constraintsFastLabel.insets = new java.awt.Insets(4, 4, 4, 4);
478     getDisplayPage().add(getFastLabel(), constraintsFastLabel);
479     // user code begin {1}

```

```

        // user code end
480    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
482        // user code end
        handleException(ivjExc);
484    }
485}
486    return ivjDisplayPage;
487}
488 /**
 * Return the DisplayQualityBox property value.
489 * @return javax.swing.JComboBox
 */
490 /* WARNING: THIS METHOD WILL BE REGENERATED. */
491 private javax.swing.JComboBox getDisplayQualityBox() {
492    if (ivjDisplayQualityBox == null) {
493        try {
494            ivjDisplayQualityBox = new javax.swing.JComboBox();
495            ivjDisplayQualityBox.setName("DisplayQualityBox");
496            // user code begin {1}
497            // user code end
498        } catch (java.lang.Throwable ivjExc) {
499            // user code begin {2}
500            // user code end
501            handleException(ivjExc);
502        }
503    }
504    return ivjDisplayQualityBox;
505}
506 /**
 * Return the DisplayTypeLabel property value.
507 * @return javax.swing.JLabel
 */
508 /* WARNING: THIS METHOD WILL BE REGENERATED. */
509 private javax.swing.JLabel getDisplayQualityLabel() {
510    if (ivjDisplayQualityLabel == null) {
511        try {
512            ivjDisplayQualityLabel = new javax.swing.JLabel();
513            ivjDisplayQualityLabel.setName("DisplayQualityLabel");
514            ivjDisplayQualityLabel.setText("DisplayQuality:");
515            // user code begin {1}
516            // user code end
517        } catch (java.lang.Throwable ivjExc) {
518            // user code begin {2}
519            // user code end
520            handleException(ivjExc);
521        }
522    }
523    return ivjDisplayQualityLabel;
524}
525 /**
526 * Return the FastLabel property value.
527 * @return javax.swing.JLabel
 */
528 /* WARNING: THIS METHOD WILL BE REGENERATED. */
529 private javax.swing.JLabel getFastLabel() {
530    if (ivjFastLabel == null) {
531        try {
532            ivjFastLabel = new javax.swing.JLabel();
533            ivjFastLabel.setName("FastLabel");
534            ivjFastLabel.setText("Fast");
535            // user code begin {1}
536            // user code end
537        } catch (java.lang.Throwable ivjExc) {
538            // user code begin {2}
539            // user code end
540        }
541    }
542    return ivjFastLabel;
543}
544
```

```

        handleException(ivjExc);
546     }
547 }
548 return ivjFastLabel;
549 */
550 /**
551 * Return the JDialoContentPane property value.
552 * @return javax.swing.JPanel
553 */
554 /* WARNING: THIS METHOD WILL BE REGENERATED. */
555 private javax.swing.JPanel getJDialoContentPane() {
556     if (ivjJDialoContentPane == null) {
557         try {
558             ivjJDialoContentPane = new javax.swing.JPanel();
559             ivjJDialoContentPane.setName("JDialoContentPane");
560             ivjJDialoContentPane.setLayout(new java.awt.BorderLayout());
561             getJDialoContentPane().add(getJTabbedPane1(), "Center");
562             getJDialoContentPane().add(get JPanel1(), "South");
563             // user code begin {1}
564             // user code end
565         } catch (java.lang.Throwable ivjExc) {
566             // user code begin {2}
567             // user code end
568             handleException(ivjExc);
569         }
570     }
571     return ivjJDialoContentPane;
572 }
573 /**
574 * Return the JPanel1 property value.
575 * @return javax.swing.JPanel
576 */
577 /* WARNING: THIS METHOD WILL BE REGENERATED. */
578 private javax.swing.JPanel get JPanel1() {
579     if (ivj JPanel1 == null) {
580         try {
581             ivj JPanel1 = new javax.swing.JPanel();
582             ivj JPanel1.setName(" JPanel1");
583             ivj JPanel1.setLayout(get JPanel1FlowLayout());
584             get JPanel1().add(getOKButton(), getOKButton().getName());
585             get JPanel1().add(getCancelButton(), getCancelButton().getName());
586             // user code begin {1}
587             // user code end
588         } catch (java.lang.Throwable ivjExc) {
589             // user code begin {2}
590             // user code end
591             handleException(ivjExc);
592         }
593     }
594     return ivj JPanel1;
595 }
596 /**
597 * Return the JPanel1FlowLayout property value.
598 * @return java.awt.FlowLayout
599 */
600 /* WARNING: THIS METHOD WILL BE REGENERATED. */
601 private java.awt.FlowLayout get JPanel1FlowLayout() {
602     java.awt.FlowLayout ivj JPanel1FlowLayout = null;
603     try {
604         /* Create part */
605         ivj JPanel1FlowLayout = new java.awt.FlowLayout();
606         ivj JPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
607     } catch (java.lang.Throwable ivjExc) {
608         handleException(ivjExc);
609     };
610     return ivj JPanel1FlowLayout;

```

```

}
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) {
619         try {
620             ivjJScrollPane1 = new javax.swing.JScrollPane();
621             ivjJScrollPane1.setName("JScrollPane1");
622             getJScrollPane1().setViewportView(getSignaturePane());
623             // user code begin {1}
624             // user code end
625         } catch (java.lang.Throwable ivjExc) {
626             // user code begin {2}
627             // user code end
628             handleException(ivjExc);
629         }
630     }
631     return ivjJScrollPane1;
632 }
633 /**
 * Return the JTabbedPane1 property value.
 * @return javax.swing.JTabbedPane
 */
635 /* WARNING: THIS METHOD WILL BE REGENERATED. */
636 private javax.swing.JTabbedPane getJTabbedPane1() {
637     if (ivjJTabbedPane1 == null) {
638         try {
639             ivjJTabbedPane1 = new javax.swing.JTabbedPane();
640             ivjJTabbedPane1.setName("JTabbedPane1");
641             ivjJTabbedPane1.insertTab("Display", null, getDisplayPage(), null, 0);
642             ivjJTabbedPane1.insertTab("Personal", null, getPersonalPage(), null, 1);
643             ivjJTabbedPane1.insertTab("Server", null, getServerPage(), null, 2);
644             // user code begin {1}
645             // user code end
646         } catch (java.lang.Throwable ivjExc) {
647             // user code begin {2}
648             // user code end
649             handleException(ivjExc);
650         }
651     }
652     return ivjJTabbedPane1;
653 }
654 /**
 * Return the JComboBox5 property value.
655 * @return javax.swing.JComboBox
 */
656 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JComboBox getLayoutSchemeBox() {
657     if (ivjLayoutSchemeBox == null) {
658         try {
659             ivjLayoutSchemeBox = new javax.swing.JComboBox();
660             ivjLayoutSchemeBox.setName("LayoutSchemeBox");
661             // user code begin {1}
662             for (int i = 0; i < ClientConstants.getLAYOUT_SCHEMES().length; i++)
663             {
664                 ivjLayoutSchemeBox.addItem( ClientConstants.getLAYOUT_SCHEMES()[i] );
665             }
666             // user code end
667         } catch (java.lang.Throwable ivjExc) {
668             // user code begin {2}
669             // user code end
670             handleException(ivjExc);
671         }
672     }
673     return ivjLayoutSchemeBox;
674 }
675

```

```

        }
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) {
687         try {
688             ivjLayoutSchemeLabel = new javax.swing.JLabel();
689             ivjLayoutSchemeLabel.setName("LayoutSchemeLabel");
690             ivjLayoutSchemeLabel.setText("LayoutScheme:");
691             // user code begin {1}
692             // user code end
693         } catch (java.lang.Throwable ivjExc) {
694             // user code begin {2}
695             // user code end
696             handleException(ivjExc);
697         }
698     }
699     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() {
707     if (ivjNameLabel == null) {
708         try {
709             ivjNameLabel = new javax.swing.JLabel();
710             ivjNameLabel.setName("NameLabel");
711             ivjNameLabel.setText("Name:");
712             // user code begin {1}
713             // user code end
714         } catch (java.lang.Throwable ivjExc) {
715             // user code begin {2}
716             // user code end
717             handleException(ivjExc);
718         }
719     }
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) {
729         try {
730             ivjNameTextField = new javax.swing.JTextField();
731             ivjNameTextField.setName("NameTextField");
732             // user code begin {1}
733             // user code end
734         } catch (java.lang.Throwable ivjExc) {
735             // user code begin {2}
736             // user code end
737             handleException(ivjExc);
738         }
739     }
740     return ivjNameTextField;
}
742 /**

```

```

    * Return the OKButton property value.
744    * @return javax.swing.JButton
745    */
746   /* WARNING: THIS METHOD WILL BE REGENERATED. */
747   private javax.swing.JButton getOKButton() {
748     if (ivjOKButton == null) {
749       try {
750         ivjOKButton = new javax.swing.JButton();
751         ivjOKButton.setName("OKButton");
752         ivjOKButton.setText("OK");
753         // user code begin {1}
754         // user code end
755         } catch (java.lang.Throwable ivjExc) {
756           // user code begin {2}
757           // user code end
758           handleException(ivjExc);
759         }
760       }
761     return ivjOKButton;
762   }
763   /**
764   * Return the PersonalPage property value.
765   * @return javax.swing.JPanel
766   */
767   /* WARNING: THIS METHOD WILL BE REGENERATED. */
768   private javax.swing.JPanel getPersonalPage() {
769     if (ivjPersonalPage == null) {
770       try {
771         ivjPersonalPage = new javax.swing.JPanel();
772         ivjPersonalPage.setName("PersonalPage");
773         ivjPersonalPage.setLayout(new java.awt.GridBagLayout());
774
775         java.awt.GridBagConstraints constraintsNameLabel = new java.awt.GridBagConstraints();
776         constraintsNameLabel.gridx = 0; constraintsNameLabel.gridy = 0;
777         constraintsNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
778         constraintsNameLabel.weighty = 0.1;
779         constraintsNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
780         getPersonalPage().add(getNameLabel(), constraintsNameLabel);
781
782         java.awt.GridBagConstraints constraintsSignatureLabel = new java.awt.GridBagConstraints();
783         constraintsSignatureLabel.gridx = 0; constraintsSignatureLabel.gridy = 1;
784         constraintsSignatureLabel.gridwidth = 2;
785         constraintsSignatureLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
786         constraintsSignatureLabel.weighty = 0.5;
787         constraintsSignatureLabel.insets = new java.awt.Insets(6, 6, 6, 6);
788         getPersonalPage().add(getSignatureLabel(), constraintsSignatureLabel);
789
790         java.awt.GridBagConstraints constraintsNameTextField = new java.awt.GridBagConstraints();
791         constraintsNameTextField.gridx = 1; constraintsNameTextField.gridy = 0;
792         constraintsNameTextField.gridwidth = 2;
793         constraintsNameTextField.fill = java.awt.GridBagConstraints.HORIZONTAL;
794         constraintsNameTextField.anchor = java.awt.GridBagConstraints.NORTHEAST;
795         constraintsNameTextField.weightx = 1.0;
796         constraintsNameTextField.weighty = 0.1;
797         constraintsNameTextField.insets = new java.awt.Insets(6, 6, 6, 6);
798         getPersonalPage().add(getNameTextField(), constraintsNameTextField);
799
800         java.awt.GridBagConstraints constraintsJScrollPane1 = new java.awt.GridBagConstraints();
801         constraintsJScrollPane1.gridx = 2; constraintsJScrollPane1.gridy = 1;
802         constraintsJScrollPane1.gridwidth = 2;
803         constraintsJScrollPane1.fill = java.awt.GridBagConstraints.BOTH;
804         constraintsJScrollPane1.anchor = java.awt.GridBagConstraints.NORTHEAST;
805         constraintsJScrollPane1.weightx = 0.5;
806         constraintsJScrollPane1.weighty = 0.5;
807         constraintsJScrollPane1.insets = new java.awt.Insets(6, 6, 6, 6);
808         getPersonalPage().add(getJScrollPane1(), constraintsJScrollPane1);

```

```

810         // user code begin {1}
811         // user code end
812     } catch (java.lang.Throwable ivjExc) {
813         // user code begin {2}
814         // user code end
815         handleException(ivjExc);
816     }
817     return ivjPersonalPage;
818 }
819 /**
820 * Return the ServerName property value.
821 * @return javax.swing.JLabel
822 */
823 /* WARNING: THIS METHOD WILL BE REGENERATED. */
824 private javax.swing.JLabel getServerName() {
825     if (ivjServerName == null) {
826         try {
827             ivjServerName = new javax.swing.JLabel();
828             ivjServerName.setName("ServerName");
829             ivjServerName.setText("Server\u00d7name:");
830             // user code begin {1}
831             // user code end
832         } catch (java.lang.Throwable ivjExc) {
833             // user code begin {2}
834             // user code end
835             handleException(ivjExc);
836         }
837     }
838     return ivjServerName;
839 }
840 /**
841 * Return the JTextField1 property value.
842 * @return javax.swing.JTextField
843 */
844 /* WARNING: THIS METHOD WILL BE REGENERATED. */
845 private javax.swing.JTextField getServerNameField() {
846     if (ivjServerNameField == null) {
847         try {
848             ivjServerNameField = new javax.swing.JTextField();
849             ivjServerNameField.setName("ServerNameField");
850             // user code begin {1}
851             // user code end
852         } catch (java.lang.Throwable ivjExc) {
853             // user code begin {2}
854             // user code end
855             handleException(ivjExc);
856         }
857     }
858     return ivjServerNameField;
859 }
860 /**
861 * Return the ServerPage property value.
862 * @return javax.swing.JPanel
863 */
864 /* WARNING: THIS METHOD WILL BE REGENERATED. */
865 private javax.swing.JPanel getServerPage() {
866     if (ivjServerPage == null) {
867         try {
868             ivjServerPage = new javax.swing.JPanel();
869             ivjServerPage.setName("ServerPage");
870             ivjServerPage.setOpaque(true);
871             ivjServerPage.setLayout(new java.awt.GridBagLayout());
872
873             java.awt.GridBagConstraints constraintsServerName = new java.awt.GridBagConstraints();
874             constraintsServerName.gridx = 0; constraintsServerName.gridy = 0;

```

```

876     constraintsServerName.anchor = java.awt.GridBagConstraints.NORTHWEST;
877     constraintsServerName.weighty = 1.0;
878     constraintsServerName.insets = new java.awt.Insets(4, 4, 4, 4);
879     getServerPage().add(getServerName(), constraintsServerName);

880     java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
881     constraintsServerNameField.gridx = 1; constraintsServerNameField.gridy = 0;
882     constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;
883     constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHWEST;
884     constraintsServerNameField.weightx = 1.0;
885     constraintsServerNameField.weighty = 1.0;
886     constraintsServerNameField.insets = new java.awt.Insets(4, 4, 4, 4);
887     getServerPage().add(getServerNameField(), constraintsServerNameField);

888     java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
889     constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
890     constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
891     constraintsServerPortLabel.weighty = 1.0;
892     constraintsServerPortLabel.insets = new java.awt.Insets(4, 4, 4, 4);
893     getServerPage().add(getServerPortLabel(), constraintsServerPortLabel);

894     java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
895     constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
896     constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
897     constraintsServerPortField.weightx = 1.0;
898     constraintsServerPortField.weighty = 1.0;
899     constraintsServerPortField.ipadx = 100;
900     constraintsServerPortField.insets = new java.awt.Insets(4, 4, 4, 4);
901     getServerPage().add(getServerPortField(), constraintsServerPortField);

902     java.awt.GridBagConstraints constraintsConnectionTypeBox = new java.awt.GridBagConstraints();
903     constraintsConnectionTypeBox.gridx = 1; constraintsConnectionTypeBox.gridy = 2;
904     constraintsConnectionTypeBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
905     constraintsConnectionTypeBox.weightx = 1.0;
906     constraintsConnectionTypeBox.insets = new java.awt.Insets(4, 4, 4, 4);
907     getServerPage().add(getConnectionTypeBox(), constraintsConnectionTypeBox);

908     java.awt.GridBagConstraints constraintsConnectionType = new java.awt.GridBagConstraints();
909     constraintsConnectionType.gridx = 0; constraintsConnectionType.gridy = 2;
910     constraintsConnectionType.insets = new java.awt.Insets(4, 4, 4, 4);
911     getServerPage().add(getConnectionType(), constraintsConnectionType);

912     java.awt.GridBagConstraints constraintsConnectButton = new java.awt.GridBagConstraints();
913     constraintsConnectButton.gridx = 1; constraintsConnectButton.gridy = 2;
914     constraintsConnectButton.anchor = java.awt.GridBagConstraints.EAST;
915     constraintsConnectButton.insets = new java.awt.Insets(4, 4, 4, 4);
916     getServerPage().add(getConnectButton(), constraintsConnectButton);
917     // user code begin {1}
918     // user code end
919   } catch (java.lang.Throwable ivjExc) {
920     // user code begin {2}
921     // user code end
922     handleException(ivjExc);
923   }
924 }
925
926 return ivjServerPage;
927 }
928 /**
929 * Return the JTextField2 property value.
930 * @return javax.swing.JTextField
931 */
932 /* WARNING: THIS METHOD WILL BE REGENERATED. */
933 private javax.swing.JTextField getServerPortField() {
934   if (ivjServerPortField == null) {
935     try {
936       ivjServerPortField = new javax.swing.JTextField();

```

```

942         ivjServerPortField.setName("ServerPortField");
943         // user code begin {1}
944         // user code end
945     } catch (java.lang.Throwable ivjExc) {
946         // user code begin {2}
947         // user code end
948         handleException(ivjExc);
949     }
950     return ivjServerPortField;
951 }
952 /**
953 * Return the ServerPortLabel property value.
954 * @return javax.swing.JLabel
955 */
956 /* WARNING: THIS METHOD WILL BE REGENERATED. */
957 private javax.swing.JLabel getServerPortLabel() {
958     if (ivjServerPortLabel == null) {
959         try {
960             ivjServerPortLabel = new javax.swing.JLabel();
961             ivjServerPortLabel.setName("ServerPortLabel");
962             ivjServerPortLabel.setText("ServerPort:");
963             // user code begin {1}
964             // user code end
965         } catch (java.lang.Throwable ivjExc) {
966             // user code begin {2}
967             // user code end
968             handleException(ivjExc);
969         }
970     }
971     return ivjServerPortLabel;
972 }
973 /**
974 * Return the SignatureLabel property value.
975 * @return javax.swing.JLabel
976 */
977 /* WARNING: THIS METHOD WILL BE REGENERATED. */
978 private javax.swing.JLabel getSignatureLabel() {
979     if (ivjSignatureLabel == null) {
980         try {
981             ivjSignatureLabel = new javax.swing.JLabel();
982             ivjSignatureLabel.setName("SignatureLabel");
983             ivjSignatureLabel.setText("Signature:");
984             // user code begin {1}
985             // user code end
986         } catch (java.lang.Throwable ivjExc) {
987             // user code begin {2}
988             // user code end
989             handleException(ivjExc);
990         }
991     }
992     return ivjSignatureLabel;
993 }
994 /**
995 * Return the JTextPane1 property value.
996 * @return javax.swing.JTextPane
997 */
998 /* WARNING: THIS METHOD WILL BE REGENERATED. */
999 private javax.swing.JTextPane getSignaturePane() {
1000     if (ivjSignaturePane == null) {
1001         try {
1002             ivjSignaturePane = new javax.swing.JTextPane();
1003             ivjSignaturePane.setName("SignaturePane");
1004             ivjSignaturePane.setBounds(0, 0, 308, 93);
1005             // user code begin {1}
1006             // user code end

```

```

    } catch (java.lang.Throwable ivjExc) {
1008        // user code begin {2}
1009        // user code end
1010        handleException(ivjExc);
    }
1012    }
1013    return ivjSignaturePane;
1014 }
1015 /**
1016 * Return the SlowLabel property value.
1017 * @return javax.swing.JLabel
1018 */
1019 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1020 private javax.swing.JLabel getSlowLabel() {
1021     if (ivjSlowLabel == null) {
1022         try {
1023             ivjSlowLabel = new javax.swing.JLabel();
1024             ivjSlowLabel.setName("SlowLabel");
1025             ivjSlowLabel.setText("Slow");
1026             // user code begin {1}
1027             // user code end
1028         } catch (java.lang.Throwable ivjExc) {
1029             // user code begin {2}
1030             // user code end
1031             handleException(ivjExc);
1032         }
1033     }
1034     return ivjSlowLabel;
1035 }
1036 /**
1037 * Return the UpdateIntervalField property value.
1038 * @return javax.swing.JTextField
1039 */
1040 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1041 private javax.swing.JTextField getUpdateIntervalField() {
1042     if (ivjUpdateIntervalField == null) {
1043         try {
1044             ivjUpdateIntervalField = new javax.swing.JTextField();
1045             ivjUpdateIntervalField.setName("UpdateIntervalField");
1046             // user code begin {1}
1047             // user code end
1048         } catch (java.lang.Throwable ivjExc) {
1049             // user code begin {2}
1050             // user code end
1051             handleException(ivjExc);
1052         }
1053     }
1054     return ivjUpdateIntervalField;
1055 }
1056 /**
1057 * Return the UpdateIntervalLabel property value.
1058 * @return javax.swing.JLabel
1059 */
1060 /* WARNING: THIS METHOD WILL BE REGENERATED. */
1061 private javax.swing.JLabel getUpdateIntervalLabel() {
1062     if (ivjUpdateIntervalLabel == null) {
1063         try {
1064             ivjUpdateIntervalLabel = new javax.swing.JLabel();
1065             ivjUpdateIntervalLabel.setName("UpdateIntervalLabel");
1066             ivjUpdateIntervalLabel.setText("Update\u00d7interval\u00d7(seconds):");
1067             // user code begin {1}
1068             // user code end
1069         } catch (java.lang.Throwable ivjExc) {
1070             // user code begin {2}
1071             // user code end
1072             handleException(ivjExc);

```

```

        }
1074     }
1075     return ivjUpdateIntervalLabel;
1076 }
1077 /**
1078  * Called whenever the part throws an exception.
1079  * @param exception java.lang.Throwable
1080  */
1081 private void handleException(java.lang.Throwable exception) {
1082     /* Uncomment the following lines to print uncaught exceptions to stdout */
1083     // System.out.println("----- UNCAUGHT EXCEPTION -----");
1084     // exception.printStackTrace(System.out);
1085 }
1086 /**
1087  * Insert the method's description here.
1088  * Creation date: (05-08-00 17:19:26)
1089  */
1090 public void init()
1091 {
1092     initialize();
1093 }
1094 /**
1095  * Initializes connections
1096  * @exception java.lang.Exception The exception description.
1097  */
1098 /**
1099  * WARNING: THIS METHOD WILL BE REGENERATED. */
1100 private void initConnections() throws java.lang.Exception {
1101     // user code begin {1}
1102     // user code end
1103     getCancelButton().addActionListener(ivjEventHandler);
1104 }
1105 /**
1106  * Initialize the class.
1107  */
1108 /**
1109  * WARNING: THIS METHOD WILL BE REGENERATED. */
1110 private void initialize() {
1111     try {
1112         // user code begin {1}
1113         // user code end
1114         setName("PreferencesDialog");
1115         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1116         setSize(509, 298);
1117         setModal(true);
1118         setTitle("Preferences");
1119         setContentPane(getJDialogContentPane());
1120         initConnections();
1121     } catch (java.lang.Throwable ivjExc) {
1122         handleException(ivjExc);
1123     }
1124     // user code begin {2}
1125     getOKButton().addActionListener( new OKHandler() );
1126     // user code end
1127 }
1128 /**
1129  * main entrypoint - starts the part when it is run as an application
1130  * @param args java.lang.String[]
1131  */
1132 public static void main(java.lang.String[] args) {
1133     try {
1134         PreferencesDialog aPreferencesDialog;
1135         aPreferencesDialog = new PreferencesDialog( new ClientApplication() );
1136         aPreferencesDialog.setModal(true);
1137         aPreferencesDialog.addWindowListener(new java.awt.event.WindowAdapter() {
1138             public void windowClosing(java.awt.event.WindowEvent e) {
1139                 System.exit(0);

```

```

        };
1140    });
1141    aPreferencesDialog.setVisible(true);
1142 } catch (Throwable exception) {
1143     System.err.println("Exception occurred in main() of javax.swing.JDialog");
1144     exception.printStackTrace(System.out);
1145 }
1146 }
1147 /**
1148 * Insert the method's description here.
1149 * Creation date: (25-06-00 20:14:31)
1150 * @param newClientApplication clientapp.ClientApplication
1151 */
1152 public void setClientApplication(ClientApplication newClientApplication) {
1153     clientApplication = newClientApplication;
1154 }
1155 /**
1156 * Overridden show method of the super-class to ensure proper initialization before the box is displayed
1157 *
1158 * Creation date: (05-08-00 20:42:55)
1159 */
1160 public void show()
1161 {
1162     if ( getConnectionTypeBox().getItemCount() == 0)
1163     {
1164         for ( int i = 0; i < ClientConstants.getConnection_TYPES().length; i++ )
1165         {
1166             getConnectionTypeBox().addItem( ClientConstants.getConnection_TYPES()[i] );
1167         }
1168     }
1169     if ( getDisplayQualityBox().getItemCount() == 0)
1170     {
1171         Iterator iterator = ClientConstants.getDISPLAY_QUALITIES().keySet().iterator();
1172         while ( iterator.hasNext() )
1173         {
1174             getDisplayQualityBox().addItem( iterator.next() );
1175         }
1176     }
1177     Properties p = ClientInfo.getPreferences();
1178     getServerNameField().setText( (String)p.getProperty( ClientConstants.getServer_NAME() ) );
1179     getServerPortField().setText( (String)p.getProperty( ClientConstants.getServer_PORT() ) );
1180     getNameTextField().setText( (String)p.getProperty( ClientConstants.getAUTHOR_NAME() ) );
1181     getSignaturePane().setText( (String)p.getProperty( ClientConstants.getSIGNATURE() ) );
1182     getLayoutSchemeBox().setSelectedItem( p.getProperty( ClientConstants.getLAYOUT_SCHEME() ) );
1183     getDisplayQualityBox().setSelectedItem( p.getProperty( ClientConstants.getDISPLAY_QUALITY() ) );
1184     getConnectionTypeBox().setSelectedItem( p.getProperty( ClientConstants.getConnection_TYPE() ) );
1185     getUpdateIntervalField().setText( (String)p.getProperty( ClientConstants.getUPDATE_INTERVAL() ) );
1186     getAnimationSpeedSlider().setValue( ClientConstants.getANIMATION_MAX_SPEED() -
1187                                         Integer.parseInt( (String)p.getProperty( ClientConstants.
1188                                         getANIMATION_DURATION_IN_MILLISECS() ) );
1189     // System.out.println( getAnimationSpeedSlider().getValue() );
1190     setLocationRelativeTo( getOwner() );
1191     super.show();
1192 }
1193 /**
1194 * Commit preferences from preferences dialog panes and dispose of the dialog box.
1195 * Creation date: (26-06-00 07:50:15)
1196 */
1197 public void submitAndClose()
1198 {
1199     Properties p = ClientInfo.getPreferences();
1200     // fetch values from dialog box and insert into Properties object
1201     if ( getDisplayQualityBox().getSelectedItem() != null )
1202     {
1203         p.setProperty( ClientConstants.getDISPLAY_QUALITY(), (String)getDisplayQualityBox().
1204                         getSelectedItem() );

```

```

1202     }
1203     if ( getLayoutSchemeBox().getSelectedItem() != null )
1204     {
1205         p.setProperty( ClientConstants.getLAYOUT_SCHEME(), (String)getLayoutSchemeBox().getSelectedItem()
1206             );
1207     }
1208     p.setProperty( ClientConstants.getAUTHOR_NAME(), (String)getNameTextField().getText() );
1209     p.setProperty( ClientConstants.getSIGNATURE(), (String)getSignaturePane().getText() );
1210     p.setProperty( ClientConstants.getServer_NAME(), (String)getServerNameField().getText() );
1211     p.setProperty( ClientConstants.getServer_PORT(), (String)getServerPortField().getText() );
1212     if ( getConnectionTypeBox().getSelectedItem() != null )
1213     {
1214         p.setProperty( ClientConstants.getConnection_TYPE(), (String)getConnectionTypeBox().
1215             getSelectedItem() );
1216     }
1217     p.setProperty( ClientConstants.getUPDATE_INTERVAL(), (String)getUpdateIntervalField().getText() );
1218     // store the inverse of the slider value to have it interpreted as a temporal value
1219     p.setProperty( ClientConstants.getANIMATION_DURATION_IN_MILLISECS(),
1220         String.valueOf( ClientConstants.getANIMATION_MAX_SPEED() - getAnimationSpeedSlider().
1221             getValue() ) );
1222     ClientInfo.setPreferences( p );
1223     ClientInfo.writePreferences();
1224     dispose();
1225 }

```

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
5     stands since the code
6     * in render() is bit of a kludge but it'll have to do for now.
7     * Creation date: (01-08-00 16:49:47)
8     * @author:
9     */
10    public class PZCamera extends edu.umd.cs.jazz.ZCamera {
11        private boolean centering = false;
12
13        /**
14         * PZCamera constructor comment.
15         */
16        public PZCamera() {
17            super();
18        }
19
20        /**
21         * PZCamera constructor comment.
22         * @param layer edu.umd.cs.jazz.ZLayerGroup
23         * @param aSurface edu.umd.cs.jazz.ZDrawingSurface
24         */
25        public PZCamera(edu.umd.cs.jazz.ZLayerGroup layer, edu.umd.cs.jazz.ZDrawingSurface aSurface) {
26            super(layer, aSurface);
27        }
28
29        /**
30         * Insert the method's description here.
31         * Creation date: (01-08-00 16:54:15)
32         * @return boolean
33         */
34        public boolean isCentering() {
35            return centering;
36        }
37
38        /**
39         * Insert the method's description here.
40         * Creation date: (01-08-00 16:56:11)
41         * @param renderContext edu.umd.cs.jazz.util.ZRenderingContext
42         */

```

```

38 public void render(edu.umd.cs.jazz.util.ZRenderingContext renderContext)
40 {
41     if ( isCentering() == true )
42     {
43         return;
44     }
45 /**
46 * Insert the method's description here.
47 * Creation date: (01-08-00 16:54:15)
48 * @param newCentering boolean
49 */
50 public void setCentering(boolean newCentering) {
51     centering = newCentering;
52 }

```

Listing C.20: PZCameraListener.java

```

package clientapp;
2
3 /**
4  * Simple extension of the JAZZ camera adapter.
5  * Creation date: (13-07-00 12:16:07)
6  * @author:
7  */
8 public class PZCameraListener extends edu.umd.cs.jazz.event.ZCameraAdapter
9 {
10 /**
11  * Insert the method's description here.
12  * Creation date: (13-07-00 12:17:43)
13  * @param cameraEvent edu.umd.cs.jazz.event.ZCameraEvent
14  */
15 public void viewChanged(edu.umd.cs.jazz.event.ZCameraEvent cameraEvent)
16 {
17
18 }

```

Listing C.21: PZoomHandler.java

```

package clientapp;
2
3 /**
4  * An extension of the JAZZ zoom handler. This solution is a bit kludgy and should be refined when
5  * convenient.
6  * Creation date: (16-06-00 23:46:26)
7  * @author:
8 */
8 public class PZoomHandler extends edu.umd.cs.jazz.event.ZoomEventHandler {
9     // kan ævre lidt suspekt at bruge en static variabel til alle swing komponenter (se pswing.render)
10    - hvad
11    // sker hvis der er flere instanser af klienten
12    private static boolean Zooming = false;
13    private static java.awt.Rectangle targetRectangle = null;
14 /**
15  * PZoomHandler constructor comment.
16  * @param node edu.umd.cs.jazz.ZNode
17  */
18 public PZoomHandler(edu.umd.cs.jazz.ZNode node) {
19     super(node);
20 }
21 /**
22  * Insert the method's description here.
23  * Creation date: (01-08-00 23:28:20)
24  * @return java.awt.Rectangle

```

```

24  /*
25   * Insert the method's description here.
26   * Creation date: (17-06-00 10:45:17)
27   * @return boolean
28   */
29  public static boolean isZooming() {
30      return Zooming;
31  }
32 /**
33  * Insert the method's description here.
34  * Creation date: (01-08-00 23:28:20)
35  * @param newTargetRectangle java.awt.Rectangle
36  */
37  public static void setTargetRectangle(java.awt.Rectangle newTargetRectangle) {
38      targetRectangle = newTargetRectangle;
39  }
40 /**
41  * Insert the method's description here.
42  * Creation date: (01-08-00 17:07:39)
43  * @param value boolean
44  */
45  public static void setZooming(boolean value)
46  {
47      Zooming = value;
48  }
49 /**
50  * Insert the method's description here.
51  * Creation date: (16-06-00 23:48:13)
52  */
53  public void startZooming()
54  {
55      // System.out.println("startZooming");
56      Zooming = true;
57      super.startZooming();
58  }
59 /**
60  * Insert the method's description here.
61  * Creation date: (17-06-00 00:59:48)
62  */
63  public void stopZooming()
64  {
65      // System.out.println("stopZooming");
66      Zooming = false;
67      super.stopZooming();
68  }
69 }

```

Listing C.22: PZPanEventHandler.java

```

1 package clientapp;
2 /**
3  * Simple extension of the JAZZ pan event handler.
4  * Creation date: (17-06-00 13:44:31)
5  * @author:
6  */
7  public class PZPanEventHandler extends edu.umd.cs.jazz.event.ZPanEventHandler {
8      private static boolean Panning = false;
9  /**
10  * PZPanEventHandler constructor comment.
11  * @param node edu.umd.cs.jazz.ZNode
12  */
13  public PZPanEventHandler(edu.umd.cs.jazz.ZNode node) {
14

```

```

        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 */
20     public PZSwing(edu.umd.cs.jazz.util.ZCanvas zbc, javax.swing.JComponent component) {
22         super(zbc, component);
    }
24 /**
* Insert the method's description here.
* Creation date: (30-07-00 17:50:17)
* @return java.awt.geom.AffineTransform
26 */
28     public java.awt.geom.AffineTransform getOriginalTransform() {
    return originalTransform;
}
30 /**
* Insert the method's description here.
32 * Creation date: (30-07-00 17:43:52)
* @return float

```

```

34  /*
35   * Insert the method's description here.
36   * Creation date: (30-07-00 17:26:06)
37   * @return edu.umd.cs.jazz.ZVisualLeaf
38  */
39  public edu.umd.cs.jazz.ZVisualLeaf getZoomingNode() {
40      return zoomingNode;
41  }
42 /**
43  * Insert the method's description here.
44  * Creation date: (07-09-00 23:31:48)
45  * @return boolean
46  */
47  public static boolean isUpdating() {
48      return updating;
49  }
50 /**
51  * Insert the method's description here.
52  * Creation date: (17-06-00 10:24:09)
53  * @param renderContext edu.umd.cs.jazz.util.ZRenderingContext
54  */
55  public void render(edu.umd.cs.jazz.util.ZRenderingContext renderContext)
56 {
57     // don't render if updating - for efficiency and safety.
58     if ( isUpdating() )
59     {
60         return;
61     }
62     // skal smukkeseres
63     if ( PZoomHandler.isZooming() || PZPanEventHandler.isPanning() )
64     {
65         paintAsGreek( renderContext.getGraphics2D() );
66         return;
67     }
68     // if a centering animation is in progress, paint as grey "blobs" rather than fully rendered graphics
69     if ( PZoomHandler.getTargetRectangle() != null )
70     {
71         // if the animation is ongoing
72         //System.out.println( renderContext.getRenderingCamera().getViewBounds() );
73         //System.out.println( PZoomHandler.getTargetRectangle() );
74         int currentWidth = (int)renderContext.getRenderingCamera().getViewBounds().getWidth();
75         int targetWidth = (int)PZoomHandler.getTargetRectangle().getWidth();
76         int currentHeight = (int)renderContext.getRenderingCamera().getViewBounds().getHeight();
77         int targetHeight = (int)PZoomHandler.getTargetRectangle().getHeight();
78         // AT ØGRE: Udskift ånedenstende magiske tal med konstant - eksperimenter for at finde en passende
79         // avrди
80         if ( (currentWidth < ( targetWidth - 3 ))|| (currentWidth > ( targetWidth + 3 ))&&
81             ( (currentHeight < ( targetHeight - 3 ))|| (currentHeight > ( targetHeight + 3 ))))
82         {
83             paintAsGreek( renderContext.getGraphics2D() );
84             return;
85         }
86     }
87     else
88     {
89         PZoomHandler.setTargetRectangle( null );
90     }
91 }
92 if ( zoomingNode instanceof edu.umd.cs.jazz.ZVisualLeaf )
93 {
94     zoomingNode.editor().getTransformGroup().scale( scaleFactor );
95 }
96 super.render(renderContext);
97 }
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 */
22 public PZText(String str) {
    super(str);
}
24 /**
  * PZText constructor comment.
26  * @param str java.lang.String
  * @param font java.awt.Font
28 */
30 public PZText(String str, java.awt.Font font) {
    super(str, font);
}
32 /**

```

```

34 * Insert the method's description here.
35 * Creation date: (16-06-00 23:07:51)
36 * @param renderContext edu.umd.cs.jazz.util.ZRenderingContext
37 */
38 public void render(edu.umd.cs.jazz.util.ZRenderingContext renderContext)
39 {
40     //renderContext.setGreekText(true);
41     super.render(renderContext);
42 }

```

Listing C.25: RemoveFiles.java

```

package clientapp;
1

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

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

16 public class RemoveFiles extends javax.swing.JDialog {

18     public class RemoveButtonHandler implements ActionListener
19     {
20         public void actionPerformed(ActionEvent actionEvent)
21         {
22             submitAndClose();
23         }
24     }

26     public class CancelButtonHandler implements ActionListener
27     {
28         public void actionPerformed(ActionEvent actionEvent)
29         {
30             dispose();
31         }
32     }

34     public class MoveLeftButtonHandler implements ActionListener
35     {
36         public void actionPerformed(ActionEvent actionEvent)
37         {
38             moveFromDisplayedToRemove();
39         }
40     }

42     public class MoveRightButtonHandler implements ActionListener
43     {
44         public void actionPerformed(ActionEvent actionEvent)
45         {
46             moveFromRemoveToDisplayed();
47         }
48     }

49     private javax.swing.JButton ivjCancelButton = null;
50     private javax.swing.JPanel ivjJDialogContentPane = null;
51     private javax.swing.JScrollPane ivjScrollPane1 = null;
52     private javax.swing.JButton ivjRemoveButton = null;
53     private javax.swing.DefaultListModel listModel = new DefaultListModel();
54     private ClientApplication clientApplication;

```

```

56     private javax.swing.JPanel ivjButtonPanel = null;
57     private javax.swing.JLabel ivjDisplayedLabel = null;
58     private javax.swing.JLabel ivjDocumentsToRemove = null;
59     private javax.swing.JScrollPane ivjJScrollPane2 = null;
60     private javax.swing.JList ivjRemoveList = null;
61     private javax.swing.JList ivjDisplayedList = null;
62     private javax.swing.JButton ivjMoveLeftButton = null;
63     private javax.swing.JButton ivjMoveRightButton = null;
64     private javax.swing.DefaultListModel removeListModel = new DefaultListModel();
65     private javax.swing.JPanel ivjBottomPanel = null;
66     private java.awt.FlowLayout ivjBottomPanelFlowLayout = null;
67     private javax.swing.JPanel ivjMainPanel = null;
68 /**
69 * RemoveFiles constructor comment.
70 */
71 public RemoveFiles() {
72     super();
73     initialize();
74 }
75 /**
76 * RemoveFiles constructor comment.
77 * @param owner java.awt.Dialog
78 */
79 public RemoveFiles(java.awt.Dialog owner) {
80     super(owner);
81 }
82 /**
83 * RemoveFiles constructor comment.
84 * @param owner java.awt.Dialog
85 * @param title java.lang.String
86 */
87 public RemoveFiles(java.awt.Dialog owner, String title) {
88     super(owner, title);
89 }
90 /**
91 * RemoveFiles constructor comment.
92 * @param owner java.awt.Dialog
93 * @param title java.lang.String
94 * @param modal boolean
95 */
96 public RemoveFiles(java.awt.Dialog owner, String title, boolean modal) {
97     super(owner, title, modal);
98 }
99 /**
100 * RemoveFiles constructor comment.
101 * @param owner java.awt.Dialog
102 * @param modal boolean
103 */
104 public RemoveFiles(java.awt.Dialog owner, boolean modal) {
105     super(owner, modal);
106 }
107 /**
108 * RemoveFiles constructor comment.
109 * @param owner java.awt.Frame
110 */
111 public RemoveFiles(java.awt.Frame owner) {
112     super(owner);
113 }
114 /**
115 * RemoveFiles constructor comment.
116 * @param owner java.awt.Frame
117 * @param title java.lang.String
118 */
119 public RemoveFiles(java.awt.Frame owner, String title) {
120     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;
189 }
190 /**
 * Return the ButtonPanel property value.
191 * @return javax.swing.JPanel
 */
194 /* WARNING: THIS METHOD WILL BE REGENERATED. */
195 private javax.swing.JPanel getButtonPanel() {
196     if (ivjButtonPanel == null) {
197         try {
198             ivjButtonPanel = new javax.swing.JPanel();
199             ivjButtonPanel.setName("ButtonPanel");
200             ivjButtonPanel.setPreferredSize(new java.awt.Dimension(31, 66));
201             ivjButtonPanel.setLayout(new java.awt.GridBagLayout());
202             ivjButtonPanel.setMinimumSize(new java.awt.Dimension(31, 66));
203             ivjButtonPanel.setMaximumSize(new java.awt.Dimension(31, 2147483647));
204
205             java.awt.GridBagConstraints constraintsMoveLeftButton = new java.awt.GridBagConstraints();
206             constraintsMoveLeftButton.gridx = 0; constraintsMoveLeftButton.gridy = 0;
207             getButtonPanel().add(getMoveLeftButton(), constraintsMoveLeftButton);
208
209             java.awt.GridBagConstraints constraintsMoveRightButton = new java.awt.GridBagConstraints();
210             constraintsMoveRightButton.gridx = 0; constraintsMoveRightButton.gridy = 1;
211             getButtonPanel().add(getMoveRightButton(), constraintsMoveRightButton);
212             // user code begin {1}
213             // user code end
214         } catch (java.lang.Throwable ivjExc) {
215             // user code begin {2}
216             // user code end
217             handleException(ivjExc);
218         }
219     }
220     return ivjButtonPanel;
221 }
222 /**
 * Return the CancelButton property value.
223 * @return javax.swing.JButton
 */
226 /* WARNING: THIS METHOD WILL BE REGENERATED. */
227 private javax.swing.JButton getCancelButton() {
228     if (ivjCancelButton == null) {
229         try {
230             ivjCancelButton = new javax.swing.JButton();
231             ivjCancelButton.setName("CancelButton");
232             ivjCancelButton.setText("Cancel");
233             // user code begin {1}
234             ivjCancelButton.addActionListener(new CancelButtonHandler());
235             // user code end
236         } catch (java.lang.Throwable ivjExc) {
237             // user code begin {2}
238             // user code end
239             handleException(ivjExc);
240         }
241     }
242     return ivjCancelButton;
243 }
244 /**
 * Insert the method's description here.
245 * Creation date: (23-06-00 00:15:50)
 * @return clientapp.ClientApplication
 */
248 public ClientApplication getClientApplication() {
249     return clientApplication;
250 }
252 /**

```

```

254 * Return the DisplayedLabel property value.
255 * @return javax.swing.JLabel
256 */
257 /* WARNING: THIS METHOD WILL BE REGENERATED. */
258 private javax.swing.JLabel getDisplayedLabel() {
259     if (ivjDisplayedLabel == null) {
260         try {
261             ivjDisplayedLabel = new javax.swing.JLabel();
262             ivjDisplayedLabel.setName("DisplayedLabel");
263             ivjDisplayedLabel.setText("Currently displayed");
264             // user code begin {1}
265             // user code end
266             } catch (java.lang.Throwable ivjExc) {
267                 // user code begin {2}
268                 // user code end
269                 handleException(ivjExc);
270             }
271         }
272     return ivjDisplayedLabel;
273 }
274 /**
275 * Return the JList1 property value.
276 * @return javax.swing.JList
277 */
278 /* WARNING: THIS METHOD WILL BE REGENERATED. */
279 private javax.swing.JList getDisplayedList() {
280     if (ivjDisplayedList == null) {
281         try {
282             ivjDisplayedList = new javax.swing.JList();
283             ivjDisplayedList.setName("DisplayedList");
284             ivjDisplayedList.setBounds(0, 0, 0, 0);
285             // user code begin {1}
286             ivjDisplayedList.setModel( getListModel() );
287             // user code end
288             } catch (java.lang.Throwable ivjExc) {
289                 // user code begin {2}
290                 // user code end
291                 handleException(ivjExc);
292             }
293         }
294     return ivjDisplayedList;
295 }
296 /**
297 * Return the DocumentsToRemove property value.
298 * @return javax.swing.JLabel
299 */
300 /* WARNING: THIS METHOD WILL BE REGENERATED. */
301 private javax.swing.JLabel getDocumentsToRemove() {
302     if (ivjDocumentsToRemove == null) {
303         try {
304             ivjDocumentsToRemove = new javax.swing.JLabel();
305             ivjDocumentsToRemove.setName("DocumentsToRemove");
306             ivjDocumentsToRemove.setText("To be removed");
307             // user code begin {1}
308             // user code end
309             } catch (java.lang.Throwable ivjExc) {
310                 // user code begin {2}
311                 // user code end
312                 handleException(ivjExc);
313             }
314         }
315     return ivjDocumentsToRemove;
316 }
317 /**
318 * Return the JDiaLogContentPane property value.
319 * @return javax.swing.JPanel

```

```

320  /*
321   * WARNING: THIS METHOD WILL BE REGENERATED.
322   */
323  private javax.swing.JPanel getJDialogContentPane() {
324      if (ivjJDialogContentPane == null) {
325          try {
326              ivjJDialogContentPane = new javax.swing.JPanel();
327              ivjJDialogContentPane.setName("JDialogContentPane");
328              ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
329              getJDialogContentPane().add(getBottomPanel(), "South");
330              getJDialogContentPane().add(getMainPanel(), "Center");
331              // user code begin {1}
332              // user code end
333          } catch (java.lang.Throwable ivjExc) {
334              // user code begin {2}
335              // user code end
336              handleException(ivjExc);
337          }
338      }
339      return ivjJDialogContentPane;
340  }
341  /**
342   * Return the JScrollPane1 property value.
343   * @return javax.swing.JScrollPane
344   */
345  /*
346   * WARNING: THIS METHOD WILL BE REGENERATED.
347   */
348  private javax.swing.JScrollPane getJScrollPane1() {
349      if (ivjJScrollPane1 == null) {
350          try {
351              ivjJScrollPane1 = new javax.swing.JScrollPane();
352              ivjJScrollPane1.setName("JScrollPane1");
353              getJScrollPane1().setViewportView(getDisplayedList());
354              // user code begin {1}
355              // user code end
356          } catch (java.lang.Throwable ivjExc) {
357              // user code begin {2}
358              // user code end
359              handleException(ivjExc);
360          }
361      }
362      return ivjJScrollPane1;
363  }
364  /**
365   * Return the JScrollPane2 property value.
366   * @return javax.swing.JScrollPane
367   */
368  /*
369   * WARNING: THIS METHOD WILL BE REGENERATED.
370   */
371  private javax.swing.JScrollPane getJScrollPane2() {
372      if (ivjJScrollPane2 == null) {
373          try {
374              ivjJScrollPane2 = new javax.swing.JScrollPane();
375              ivjJScrollPane2.setName("JScrollPane2");
376              getJScrollPane2().setViewportView(getRemoveList());
377              // user code begin {1}
378              // user code end
379          } catch (java.lang.Throwable ivjExc) {
380              // user code begin {2}
381              // user code end
382              handleException(ivjExc);
383          }
384      }
385      return ivjJScrollPane2;
386  }
387  /**
388   * Insert the method's description here.
389   * Creation date: (22-06-00 23:40:02)
390   * @return javax.swing.DefaultListModel

```

```

        */
386 public javax.swing.DefaultListModel getListModel() {
    return listModel;
388 }
389 /**
390 * Return the JPanel2 property value.
391 * @return javax.swing.JPanel
392 */
393 /* WARNING: THIS METHOD WILL BE REGENERATED. */
394 private javax.swing.JPanel getMainPanel() {
    if (ivjMainPanel == null) {
395     try {
396         ivjMainPanel = new javax.swing.JPanel();
397         ivjMainPanel.setName("MainPanel");
398         ivjMainPanel.setLayout(new java.awt.GridBagLayout());
399
400         java.awt.GridBagConstraints constraintsButtonPanel = new java.awt.GridBagConstraints();
401         constraintsButtonPanel.gridx = 1; constraintsButtonPanel.gridy = 1;
402         constraintsButtonPanel.fill = java.awt.GridBagConstraints.VERTICAL;
403         getMainPanel().add(getButtonPanel(), constraintsButtonPanel);
404
405         java.awt.GridBagConstraints constraintsDisplayedLabel = new java.awt.GridBagConstraints();
406         constraintsDisplayedLabel.gridx = 0; constraintsDisplayedLabel.gridy = 0;
407         constraintsDisplayedLabel.anchor = java.awt.GridBagConstraints.WEST;
408         constraintsDisplayedLabel.insets = new java.awt.Insets(4, 4, 4, 4);
409         getMainPanel().add(getDisplayedLabel(), constraintsDisplayedLabel);
410
411         java.awt.GridBagConstraints constraintsJScrollPane1 = new java.awt.GridBagConstraints();
412         constraintsJScrollPane1.gridx = 0; constraintsJScrollPane1.gridy = 1;
413         constraintsJScrollPane1.fill = java.awt.GridBagConstraints.BOTH;
414         constraintsJScrollPane1.anchor = java.awt.GridBagConstraints.WEST;
415         constraintsJScrollPane1.weightx = 1.0;
416         constraintsJScrollPane1.weighty = 1.0;
417         constraintsJScrollPane1.insets = new java.awt.Insets(4, 4, 4, 4);
418         getMainPanel().add(getJScrollPane1(), constraintsJScrollPane1);
419
420         java.awt.GridBagConstraints constraintsJScrollPane2 = new java.awt.GridBagConstraints();
421         constraintsJScrollPane2.gridx = 2; constraintsJScrollPane2.gridy = 1;
422         constraintsJScrollPane2.fill = java.awt.GridBagConstraints.BOTH;
423         constraintsJScrollPane2.anchor = java.awt.GridBagConstraints.EAST;
424         constraintsJScrollPane2.weightx = 1.0;
425         constraintsJScrollPane2.weighty = 1.0;
426         constraintsJScrollPane2.insets = new java.awt.Insets(4, 4, 4, 4);
427         getMainPanel().add(getJScrollPane2(), constraintsJScrollPane2);
428
429         java.awt.GridBagConstraints constraintsDocumentsToRemove = new java.awt.GridBagConstraints();
430         constraintsDocumentsToRemove.gridx = 2; constraintsDocumentsToRemove.gridy = 0;
431         constraintsDocumentsToRemove.anchor = java.awt.GridBagConstraints.WEST;
432         constraintsDocumentsToRemove.insets = new java.awt.Insets(4, 4, 4, 4);
433         getMainPanel().add(getDocumentsToRemove(), constraintsDocumentsToRemove);
434         // user code begin {1}
435         getMainPanel().setBorder( ClientConstants.createDialogBorder( getMainPanel().getBackground() ) );
436         // user code end
437     } catch (java.lang.Throwable ivjExc) {
438         // user code begin {2}
439         // user code end
440         handleException(ivjExc);
441     }
442 }
443 return ivjMainPanel;
444 }
445 /**
446 * Return the AddButton property value.
447 * @return javax.swing.JButton
448 */
449 /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

    private javax.swing.JButton getMoveLeftButton() {
452     if (ivjMoveLeftButton == null) {
453         try {
454             ivjMoveLeftButton = new javax.swing.JButton();
455             ivjMoveLeftButton.setName("MoveLeftButton");
456             ivjMoveLeftButton.setToolTipText("Add_to_list");
457             ivjMoveLeftButton.setMnemonic('a');
458             ivjMoveLeftButton.setText("");
459             ivjMoveLeftButton.setIcon(new javax.swing.ImageIcon(getClass().getResource(
460                     "toolbarButtonGraphics/navigation/Forward24.gif")));
461             ivjMoveLeftButton.setBorderPainted(false);
462             ivjMoveLeftButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
463             // user code begin {1}
464             ivjMoveLeftButton.addActionListener(new MoveLeftButtonHandler());
465             // user code end
466         } catch (java.lang.Throwable ivjExc) {
467             // user code begin {2}
468             // user code end
469             handleException(ivjExc);
470         }
471     }
472     return ivjMoveLeftButton;
473 }
474 /**
475 * Return the RemoveButton1 property value.
476 * @return javax.swing.JButton
477 */
478 /* WARNING: THIS METHOD WILL BE REGENERATED. */
479 private javax.swing.JButton getMoveRightButton() {
480     if (ivjMoveRightButton == null) {
481         try {
482             ivjMoveRightButton = new javax.swing.JButton();
483             ivjMoveRightButton.setName("MoveRightButton");
484             ivjMoveRightButton.setToolTipText("Remove_from_list");
485             ivjMoveRightButton.setMnemonic('r');
486             ivjMoveRightButton.setText("");
487             ivjMoveRightButton.setMaximumSize(new java.awt.Dimension(29, 35));
488             ivjMoveRightButton.setIcon(new javax.swing.ImageIcon(getClass().getResource(
489                     "toolbarButtonGraphics/navigation/Back24.gif")));
490             ivjMoveRightButton.setBorderPainted(false);
491             ivjMoveRightButton.setPreferredSize(new java.awt.Dimension(29, 35));
492             ivjMoveRightButton.setMinimumSize(new java.awt.Dimension(29, 35));
493             ivjMoveRightButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
494             // user code begin {1}
495             ivjMoveRightButton.addActionListener(new MoveRightButtonHandler());
496             // user code end
497         } catch (java.lang.Throwable ivjExc) {
498             // user code begin {2}
499             // user code end
500             handleException(ivjExc);
501         }
502     }
503     return ivjMoveRightButton;
504 }
505 /**
506 * Return the RemoveButton property value.
507 * @return javax.swing.JButton
508 */
509 /* WARNING: THIS METHOD WILL BE REGENERATED. */
510 private javax.swing.JButton getRemoveButton() {
511     if (ivjRemoveButton == null) {
512         try {
513             ivjRemoveButton = new javax.swing.JButton();
514             ivjRemoveButton.setName("RemoveButton");
515             ivjRemoveButton.setMnemonic('r');
516             ivjRemoveButton.setText("Remove");
517         }
518     }
519     return ivjRemoveButton;
520 }

```

```

516         // user code begin {1}
517         ivjRemoveButton.addActionListener(new RemoveButtonHandler() );
518     } catch (java.lang.Throwable ivjExc) {
519         // user code begin {2}
520         // user code end
521         handleException(ivjExc);
522     }
523 }
524 return ivjRemoveButton;
}
526 /**
527 * Return the RemoveList property value.
528 * @return javax.swing.JList
529 */
530 /* WARNING: THIS METHOD WILL BE REGENERATED. */
531 private javax.swing.JList getRemoveList() {
532     if (ivjRemoveList == null) {
533         try {
534             ivjRemoveList = new javax.swing.JList();
535             ivjRemoveList.setName("RemoveList");
536             ivjRemoveList.setBackground(java.awt.Color.white);
537             ivjRemoveList.setBounds(0, 0, 160, 120);
538             // user code begin {1}
539             ivjRemoveList.setModel(removeListModel);
540             // user code end
541         } catch (java.lang.Throwable ivjExc) {
542             // user code begin {2}
543             // user code end
544             handleException(ivjExc);
545         }
546     }
547     return ivjRemoveList;
548 }
549 /**
550 * Insert the method's description here.
551 * Creation date: (23-06-00 19:55:23)
552 * @return javax.swing.DefaultListModel
553 */
554 public javax.swing.DefaultListModel getRemoveListModel() {
555     return removeListModel;
556 }
557 /**
558 * Called whenever the part throws an exception.
559 * @param exception java.lang.Throwable
560 */
561 private void handleException(java.lang.Throwable exception) {
562     /* Uncomment the following lines to print uncaught exceptions to stdout */
563     // System.out.println("----- UNCAUGHT EXCEPTION -----");
564     // exception.printStackTrace(System.out);
565 }
566 /**
567 * Insert the method's description here.
568 * Creation date: (22-06-00 23:11:53)
569 */
570 public void initialize() {
571     try {
572         // user code begin {1}
573         // user code end
574         setName("RemoveFiles");
575         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
576         setSize(600, 300);
577         setTitle("RemoveDocuments");
578         setContentPane(getJDialogContentPane());
579     } catch (java.lang.Throwable ivjExc) {
580

```

```

        handleException(ivjExc);
582    }
583    // user code begin {2}
584    getJDialogContentPane().setBorder( javax.swing.BorderFactory.createMatteBorder(6,6,6,6,
585                                    getJDialogContentPane().getBackground() ) );
586    // user code end
587 }
588 /**
589 * main entrypoint - starts the part when it is run as an application
590 * @param args java.lang.String[]
591 */
592 public static void main(java.lang.String[] args) {
593     try {
594         RemoveFiles aRemoveFiles;
595         aRemoveFiles = new RemoveFiles();
596         aRemoveFiles.setModal(true);
597         aRemoveFiles.addWindowListener(new java.awt.event.WindowAdapter() {
598             public void windowClosing(java.awt.event.WindowEvent e) {
599                 System.exit(0);
600             };
601         });
602         aRemoveFiles.setVisible(true);
603     } catch (Throwable exception) {
604         System.err.println("Exception occurred in main() of javax.swing.JDialog");
605         exception.printStackTrace(System.out);
606     }
607 }
608 /**
609 * Insert the method's description here.
610 * Creation date: (23-06-00 20:04:34)
611 */
612 public void moveFromDisplayedToRemove()
613 {
614     for (int i = listModel.size()-1; i >= 0; i--)
615     {
616         if ( getDisplayedList().isSelectedIndex(i) )
617         {
618             removeListModel.addElement( listModel.get(i) );
619             listModel.remove(i);
620         }
621     }
622 }
623 /**
624 * Insert the method's description here.
625 * Creation date: (23-06-00 20:05:56)
626 */
627 public void moveFromRemoveToDisplayed()
628 {
629     for (int i = removeListModel.size()-1; i >= 0; i--)
630     {
631         if ( getRemoveList().isSelectedIndex(i) )
632         {
633             listModel.addElement( removeListModel.get(i) );
634             removeListModel.remove(i);
635         }
636     }
637 }
638 /**
639 * Insert the method's description here.
640 * Creation date: (23-06-00 00:15:50)
641 * @param newClientApplication clientapp.ClientApplication
642 */
643 public void setClientApplication(ClientApplication newClientApplication) {
644     clientApplication = newClientApplication;
645 }
646 /**

```

```

646 * Insert the method's description here.
647 * Creation date: (22-06-00 23:40:02)
648 * @param newListModel javax.swing.DefaultListModel
649 */
650 public void setListModel(javax.swing.DefaultListModel newListModel) {
651     listModel = newListModel;
652 }
653 /**
654 * Insert the method's description here.
655 * Creation date: (23-06-00 19:55:23)
656 * @param newRemoveListModel javax.swing.DefaultListModel
657 */
658 public void setRemoveListModel(javax.swing.DefaultListModel newRemoveListModel) {
659     removeListModel = newRemoveListModel;
660 }
661 /**
662 * Insert the method's description here.
663 * Creation date: (19-07-00 02:38:38)
664 */
665 public void show()
666 {
667     listModel.clear();
668     removeListModel.clear();
669     fillListBox( clientApplication.getClientManager().getClientInfo().getLocalFiles() );
670     // default positioning puts the dialog at center of screen. The method used is a slight hack, but
671     // works fine.
672     setLocationRelativeTo( this );
673     setLocation( (int) getLocation().getX(), (int) getLocation().getY() );
674     super.show();
675 }
676 /**
677 * Insert the method's description here.
678 * Creation date: (23-06-00 00:12:51)
679 */
680 public void submitAndClose()
681 {
682     if ( removeListModel.size() > 0 )
683     {
684         File removables[] = new File[removeListModel.size()];
685         File remainingFiles[] = new File[listModel.size()];
686         for (int i = 0; i < removables.length; i++)
687         {
688             removables[i] = (File)removeListModel.get(i);
689         }
690         clientApplication.getClientManager().removeDocuments( removables );
691     }
692     else
693     {
694         clientApplication.initMessageArea( 1, 1, 1, ClientConstants.getNOTHING_TO_REMOVE_MESSAGE() );
695     }
696     dispose();
697 }

```

Listing C.26: Validator.java

```

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

```

```
12  /**
13   * Validator constructor comment.
14   */
15  public Validator( ClientApplication clientApplication )
16  {
17      super();
18      setClientApplication( clientApplication );
19  }
20  /**
21   * Insert the method's description here.
22   * Creation date: (26-07-00 16:07:18)
23   * @return clientapp.ClientApplication
24   */
25  public ClientApplication getClientApplication() {
26      return ClientApplication;
27  }
28  /**
29   * Insert the method's description here.
30   * Creation date: (26-07-00 16:07:18)
31   * @param newClientApplication clientapp.ClientApplication
32   */
33  public void setClientApplication(ClientApplication newClientApplication) {
34      ClientApplication = newClientApplication;
35  }
36 }
```

Listing C.27: GroupManagementServer.java

```
1 package serverapp;
2
3 import com.sun.media.jsdt.*;
4 /**
5  * This class is intended for use by PeerView servers connecting to a JSDT session.
6  * Creation date: (03-07-00 12:17:46)
7  * @author:
8  */
9 public class GroupManagementServer implements Client, ChannelConsumer {
10     private java.lang.String name = "";
11 /**
12  * GroupManagementServer constructor comment.
13  */
14 public GroupManagementServer() {
15     super();
16 }
17 /**
18  * Insert the method's description here.
19  * Creation date: (03-07-00 12:30:58)
20  * @param ai com.sun.media.jsdt.AuthenticationInfo
21  */
22 public Object authenticate(AuthenticationInfo ai)
23 {
24     return null;
25 }
26 /**
27  * dataReceived method comment.
28  */
29 public synchronized void dataReceived(Data data)
30 {
31
32 }
33 /**
34  * Insert the method's description here.
35  * Creation date: (03-07-00 12:21:42)
36  * @return java.lang.String
37  */
38 public java.lang.String getName() {
```

```

40     return name;
41 }
42 /**
43 * Insert the method's description here.
44 * Creation date: (03-07-00 12:21:42)
45 * @param newName java.lang.String
46 */
47 public void setName(java.lang.String newName) {
48     name = newName;
49 }
50 }

```

Listing C.28: PeerViewServer.java

```

package serverapp;
1
2 import com.sun.media.jsdt.*;
3 /**
4  * The PeerView server class.
5  * Creation date: (30-06-00 10:00:51)
6  * @author:
7  */
8
9 public class PeerViewServer extends ServerInfo implements Client, ChannelConsumer {
10    private java.lang.String name;
11 /**
12  * PeerViewServer constructor comment.
13  */
14 public PeerViewServer(String nameArg)
15 {
16    super();
17    name = nameArg;
18 }
19 /**
20  * authenticate method comment.
21  */
22 public Object authenticate(AuthenticationInfo arg1) {
23    return null;
24 }
25 /**
26  * Insert the method's description here.
27  * Creation date: (30-06-00 14:24:04)
28  * @param data com.sun.media.jsdt.Data
29  */
30 public synchronized void dataReceived(Data data)
31 {
32    // System.out.println( data.getDataAsString() );
33 }
34 /**
35  * Insert the method's description here.
36  * Creation date: (30-06-00 10:15:32)
37  * @return java.lang.String
38  */
39 public java.lang.String getName() {
40    return name;
41 }
42 /**
43  * Insert the method's description here.
44  * Creation date: (30-06-00 10:15:32)
45  * @param newName java.lang.String
46  */
47 public void setName(java.lang.String newName) {
48    name = newName;
49 }
50 }

```

Listing C.29: ServerApplication.java

```

package serverapp;
2   import com.sun.media.jsdt.*;
4   import peerviewmisc.*;
import java.util.*;
6   import java.awt.Dimension;
import java.awt.Point;
8
 /**
10  * This class implements the server application main window.
11  * Creation date: (29-06-00 17:57:51)
12  * @author:
13  */
14 public class ServerApplication extends javax.swing.JFrame {
    public Channel channel;
16  private peerviewmisc.AboutDialog ivjAboutDialog = null;
    private javax.swing.JMenuItem ivjAboutMenuItem = null;
18  private javax.swing.JMenuItem ivjContentsMenuItem = null;
    IvjEventHandler ivjEventHandler = new IvjEventHandler();
20  private javax.swing.JButton ivjExitButton = null;
    private javax.swing.JMenuItem ivjExitMenuItem = null;
22  private javax.swing.JButton ivjHelpButton = null;
    private javax.swing.JMenu ivjHelpMenu = null;
24  private javax.swing.JPanel ivjJFrameContentPane = null;
    private javax.swing.JPanel ivjJPanel1 = null;
26  private javax.swing.JScrollPane ivj JScrollPane1 = null;
    private javax.swing.JSeparator ivjJSeparator1 = null;
28  private javax.swing.JToolBar ivjJToolBar1 = null;
    private javax.swing.JTable ivjScrollPaneTable = null;
30  private javax.swing.JMenuBar ivjServerApplicationJMenuBar = null;
    private javax.swing.JMenu ivjServerMenu = null;
32  private ServerSetup ivjServerSetup = null;
    private javax.swing.JButton ivjSetupButton = null;
34  private javax.swing.JMenuItem ivjSetupMenuItem = null;
    private ServerManager serverManager = null;
36  private javax.swing.table.DefaultTableModel tableModel = null;
    private HelpDialog ivjHelpDialog1 = null;
38
    class IvjEventHandler implements java.awt.event.ActionListener {
40        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);
        };
58    };
    /**
60     * ServerApplication constructor comment.
61     */
62 public ServerApplication() {
    super();
64     initialize();
}
66 /**

```

```

    * ServerApplication constructor comment.
68    * @param title java.lang.String
    */
70   public ServerApplication(String title) {
    super(title);
72 }
73 /**
74  * Center and size the window at predefined dimensions and position relative to the screen resolution
75  * Creation date: (05-08-00 19:20:24)
76  */
77   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() ));
85   Point position;
86   Point preferredPosition = new Point(
        Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
            getFRAME_X_COORDINATE() )),
        Integer.parseInt( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
            getFRAME_Y_COORDINATE() )));
87   Point defaultPosition = new Point(
        Integer.parseInt( ServerConstants.getDEFAULT_FRAME_X_COORDINATE() ),
        Integer.parseInt( ServerConstants.getDEFAULT_FRAME_Y_COORDINATE() ));
88 // if no value has overridden the default setting then query the OS for the dimensions of the display
89   if ( preferredSize.equals( defaultSize ) )
90   {
91     size = new Dimension( (int) (getToolkit().getScreenSize().getWidth() * ServerConstants.
92         getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()),
93           (int) (getToolkit().getScreenSize().getHeight() * ServerConstants.
94             getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() ) );
95   }
96   else
97   {
98     size = preferredSize;
99   }
100  if ( preferredPosition.equals( defaultPosition ) )
101  {
102    position = new Point( (int)( getToolkit().getScreenSize().getWidth() * ( (1.0 - ServerConstants.
103        getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) /2 ) ),
104      (int)( getToolkit().getScreenSize().getHeight() * ( (1.0 - ServerConstants.
105          getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE()) /2 ) );
106  }
107  else
108  {
109    position = preferredPosition;
110  }
111 // System.out.println( size.toString() );
112  setSize( size );
113  setLocation( position );
114  setState( java.awt.Frame.NORMAL );
115  repaint();
116 }
117 /**
118  * connEtoM1: (SetupMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> ServerSetup.show()V)
119  * @param arg1 java.awt.event.ActionEvent
120  */
121 /* WARNING: THIS METHOD WILL BE REGENERATED. */
122 private void connEtoM1(java.awt.event.ActionEvent arg1) {
123   try {

```

```

126     // user code begin {1}
127     // user code end
128     getServerSetup().show();
129     // user code begin {2}
130     // user code end
131 } catch (java.lang.Throwable ivjExc) {
132     // user code begin {3}
133     // user code end
134     handleException(ivjExc);
135 }
136 /**
137 * connEtoM2: (ExitMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.
138 *             dispose()V)
139 * @param arg1 java.awt.event.ActionEvent
140 */
141 /* WARNING: THIS METHOD WILL BE REGENERATED. */
142 private void connEtoM2(java.awt.event.ActionEvent arg1) {
143     try {
144         // user code begin {1}
145         // user code end
146         this.terminateAndClose();
147         // user code begin {2}
148         // user code end
149     } catch (java.lang.Throwable ivjExc) {
150         // user code begin {3}
151         // user code end
152         handleException(ivjExc);
153     }
154 }
155 /**
156 * connEtoM3: (AboutMenuItem.action.actionPerformed(java.awt.event.ActionEvent) --> AboutDialog.show()V)
157 * @param arg1 java.awt.event.ActionEvent
158 */
159 /* WARNING: THIS METHOD WILL BE REGENERATED. */
160 private void connEtoM3(java.awt.event.ActionEvent arg1) {
161     try {
162         // user code begin {1}
163         // user code end
164         getAboutDialog().show();
165         // user code begin {2}
166         // user code end
167     } catch (java.lang.Throwable ivjExc) {
168         // user code begin {3}
169         // user code end
170         handleException(ivjExc);
171     }
172 }
173 /**
174 * connEtoM4: (NewGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.show()V)
175 * @param arg1 java.awt.event.ActionEvent
176 */
177 /* WARNING: THIS METHOD WILL BE REGENERATED. */
178 private void connEtoM4(java.awt.event.ActionEvent arg1) {
179     try {
180         // user code begin {1}
181         // user code end
182         this.dispose();
183         // user code begin {2}
184         // user code end
185     } catch (java.lang.Throwable ivjExc) {
186         // user code begin {3}
187         // user code end
188         handleException(ivjExc);
189     }
190 }
```

```

190 /**
191 * connEtoM5: (EditGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> EditGroup1.show()V)
192 * @param arg1 java.awt.event.ActionEvent
193 */
194 /* WARNING: THIS METHOD WILL BE REGENERATED. */
195 private void connEtoM5(java.awt.event.ActionEvent arg1) {
196     try {
197         // user code begin {1}
198         // user code end
199         getHelpDialog1().show();
200         // user code begin {2}
201         // user code end
202     } catch (java.lang.Throwable ivjExc) {
203         // user code begin {3}
204         // user code end
205         handleException(ivjExc);
206     }
207 }
208 /**
209 * connEtoM6: (DeleteGroupItem.action.actionPerformed(java.awt.event.ActionEvent) --> DeleteGroup1.show()
210 * @param arg1 java.awt.event.ActionEvent
211 */
212 /* WARNING: THIS METHOD WILL BE REGENERATED. */
213 private void connEtoM6(java.awt.event.ActionEvent arg1) {
214     try {
215         // user code begin {1}
216         // user code end
217         getHelpDialog1().show();
218         // user code begin {2}
219         // user code end
220     } catch (java.lang.Throwable ivjExc) {
221         // user code begin {3}
222         // user code end
223         handleException(ivjExc);
224     }
225 }
226 /**
227 * connEtoM7: (DefaultToolBarButton.action.actionPerformed(java.awt.event.ActionEvent) --> NewGroup1.
228 * show()V)
229 * @param arg1 java.awt.event.ActionEvent
230 */
231 /* WARNING: THIS METHOD WILL BE REGENERATED. */
232 private void connEtoM7(java.awt.event.ActionEvent arg1) {
233     try {
234         // user code begin {1}
235         // user code end
236         getServerSetup().show();
237         // user code begin {2}
238         // user code end
239     } catch (java.lang.Throwable ivjExc) {
240         // user code begin {3}
241         // user code end
242         handleException(ivjExc);
243     }
244 }
245 /**
246 * connEtoM8: (ExitButton.action.actionPerformed(java.awt.event.ActionEvent) --> ServerApplication.
247 * terminateAndClose()V)
248 * @param arg1 java.awt.event.ActionEvent
249 */
250 /* WARNING: THIS METHOD WILL BE REGENERATED. */
251 private void connEtoM8(java.awt.event.ActionEvent arg1) {
252     try {
253         // user code begin {1}
254         // user code end

```

```

        this.terminateAndClose();
254    // user code begin {2}
255    // user code end
256 } catch (java.lang.Throwable ivjExc) {
257     // user code begin {3}
258     // user code end
259     handleException(ivjExc);
260 }
261 /**
262  * Insert the method's description here.
263  * Creation date: (07-07-00 01:53:28)
264  */
265 protected void finalize() throws Throwable
266 {
267     terminateAndClose();
268     super.finalize();
269 }
270 /**
271  * Return the AboutDialog property value.
272  * @return clientapp.AboutDialog
273  */
274 /* WARNING: THIS METHOD WILL BE REGENERATED. */
275 private peerviewmisc.AboutDialog getAboutDialog() {
276     if (ivjAboutDialog == null) {
277         try {
278             ivjAboutDialog = new peerviewmisc.AboutDialog();
279             ivjAboutDialog.setName("AboutDialog");
280             ivjAboutDialog.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
281             // user code begin {1}
282             // user code end
283         } catch (java.lang.Throwable ivjExc) {
284             // user code begin {2}
285             // user code end
286             handleException(ivjExc);
287         }
288     }
289     return ivjAboutDialog;
290 }
291 /**
292  * Return the AboutMenuItem property value.
293  * @return javax.swing.JMenuItem
294  */
295 /* WARNING: THIS METHOD WILL BE REGENERATED. */
296 private javax.swing.JMenuItem getAboutMenuItem() {
297     if (ivjAboutMenuItem == null) {
298         try {
299             ivjAboutMenuItem = new javax.swing.JMenuItem();
300             ivjAboutMenuItem.setName("AboutMenuItem");
301             ivjAboutMenuItem.setText("About");
302             // user code begin {1}
303             // user code end
304         } catch (java.lang.Throwable ivjExc) {
305             // user code begin {2}
306             // user code end
307             handleException(ivjExc);
308         }
309     }
310     return ivjAboutMenuItem;
311 }
312 /**
313  * Return the ContentsMenuItem property value.
314  * @return javax.swing.JMenuItem
315  */
316 /* WARNING: THIS METHOD WILL BE REGENERATED. */
317 private javax.swing.JMenuItem getContentsMenuItem() {

```

```

320     if (ivjContentsMenuItem == null) {
321         try {
322             ivjContentsMenuItem = new javax.swing.JMenuItem();
323             ivjContentsMenuItem.setName("ContentsMenuItem");
324             ivjContentsMenuItem.setText("Contents");
325             // user code begin {1}
326             // user code end
327         } catch (java.lang.Throwable ivjExc) {
328             // user code begin {2}
329             // user code end
330             handleException(ivjExc);
331         }
332     }
333 }
334 /**
335 * Return the ExitButton property value.
336 * @return javax.swing.JButton
337 */
338 /* WARNING: THIS METHOD WILL BE REGENERATED. */
339 private javax.swing.JButton getExitButton() {
340     if (ivjExitButton == null) {
341         try {
342             ivjExitButton = new javax.swing.JButton();
343             ivjExitButton.setName("ExitButton");
344             ivjExitButton.setToolTipText("Exit");
345             ivjExitButton.setText("");
346             ivjExitButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
347             ivjExitButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
348             ivjExitButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
349                 general/Stop24.gif")));
350             ivjExitButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
351             // user code begin {1}
352             // user code end
353         } catch (java.lang.Throwable ivjExc) {
354             // user code begin {2}
355             // user code end
356             handleException(ivjExc);
357         }
358     }
359     return ivjExitButton;
360 }
361 /**
362 * Return the ExitMenuItem property value.
363 * @return javax.swing.JMenuItem
364 */
365 /* WARNING: THIS METHOD WILL BE REGENERATED. */
366 private javax.swing.JMenuItem getExitMenuItem() {
367     if (ivjExitMenuItem == null) {
368         try {
369             ivjExitMenuItem = new javax.swing.JMenuItem();
370             ivjExitMenuItem.setName("ExitMenuItem");
371             ivjExitMenuItem.setText("Exit");
372             // user code begin {1}
373             // user code end
374         } catch (java.lang.Throwable ivjExc) {
375             // user code begin {2}
376             // user code end
377             handleException(ivjExc);
378         }
379     }
380     return ivjExitMenuItem;
381 }
382 /**
383 * Return the HelpButton property value.
384 * @return javax.swing.JButton

```

```

384  /*
385   * WARNING: THIS METHOD WILL BE REGENERATED. */
386  private javax.swing.JButton getHelpButton() {
387      if (ivjHelpButton == null) {
388          try {
389              ivjHelpButton = new javax.swing.JButton();
390              ivjHelpButton.setName("HelpButton");
391              ivjHelpButton.setToolTipText("Help");
392              ivjHelpButton.setText("");
393              ivjHelpButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
394              ivjHelpButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
395              ivjHelpButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
396                  general/Help24.gif")));
397              ivjHelpButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
398              // user code begin {1}
399              // user code end
400          } catch (java.lang.Throwable ivjExc) {
401              // user code begin {2}
402              // user code end
403              handleException(ivjExc);
404          }
405      }
406      return ivjHelpButton;
407  }
408 /**
409  * Return the HelpDialog1 property value.
410  * @return peerviewmisc.HelpDialog
411  */
412 /* WARNING: THIS METHOD WILL BE REGENERATED. */
413 private peerviewmisc.HelpDialog getHelpDialog1() {
414     if (ivjHelpDialog1 == null) {
415         try {
416             ivjHelpDialog1 = new peerviewmisc.HelpDialog();
417             ivjHelpDialog1.setName("HelpDialog1");
418             ivjHelpDialog1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
419             // user code begin {1}
420             // user code end
421         } catch (java.lang.Throwable ivjExc) {
422             // user code begin {2}
423             // user code end
424             handleException(ivjExc);
425         }
426     }
427     return ivjHelpDialog1;
428 }
429 /**
430  * Return the HelpMenu property value.
431  * @return javax.swing.JMenu
432  */
433 /* WARNING: THIS METHOD WILL BE REGENERATED. */
434 private javax.swing.JMenu getHelpMenu() {
435     if (ivjHelpMenu == null) {
436         try {
437             ivjHelpMenu = new javax.swing.JMenu();
438             ivjHelpMenu.setName("HelpMenu");
439             ivjHelpMenu.setText("Help");
440             ivjHelpMenu.add(getContentsMenuItem());
441             ivjHelpMenu.add(getAboutMenuItem());
442             // user code begin {1}
443             // user code end
444         } catch (java.lang.Throwable ivjExc) {
445             // user code begin {2}
446             // user code end
447             handleException(ivjExc);
448         }
449     }

```

```

        return ivjHelpMenu;
450 }
451 /**
452 * Return the JFrameContentPane property value.
453 * @return javax.swing.JPanel
454 */
455 /* WARNING: THIS METHOD WILL BE REGENERATED. */
456 private javax.swing.JPanel getJFrameContentPane() {
457     if (ivjJFrameContentPane == null) {
458         try {
459             ivjJFrameContentPane = new javax.swing.JPanel();
460             ivjJFrameContentPane.setName("JFrameContentPane");
461             ivjJFrameContentPane.setLayout(new java.awt.BorderLayout());
462             getJFrameContentPane().add(getJToolBar1(), "North");
463             getJFrameContentPane().add(getJPanel1(), "Center");
464             // user code begin {1}
465             // user code end
466         } catch (java.lang.Throwable ivjExc) {
467             // user code begin {2}
468             // user code end
469             handleException(ivjExc);
470         }
471     }
472     return ivjJFrameContentPane;
473 }
474 /**
475 * Return the JPanel1 property value.
476 * @return javax.swing.JPanel
477 */
478 /* WARNING: THIS METHOD WILL BE REGENERATED. */
479 private javax.swing.JPanel getJPanel1() {
480     if (ivjJPanel1 == null) {
481         try {
482             ivjJPanel1 = new javax.swing.JPanel();
483             ivjJPanel1.setName("JPanel1");
484             ivjJPanel1.setLayout(new java.awt.BorderLayout());
485             getJPanel1().add(getJScrollPane1(), "Center");
486             // user code begin {1}
487             // user code end
488         } catch (java.lang.Throwable ivjExc) {
489             // user code begin {2}
490             // user code end
491             handleException(ivjExc);
492         }
493     }
494     return ivjJPanel1;
495 }
496 /**
497 * Return the JScrollPane1 property value.
498 * @return javax.swing.JScrollPane
499 */
500 /* WARNING: THIS METHOD WILL BE REGENERATED. */
501 private javax.swing.JScrollPane getJScrollPane1() {
502     if (ivjJScrollPane1 == null) {
503         try {
504             ivjJScrollPane1 = new javax.swing.JScrollPane();
505             ivjJScrollPane1.setName("JScrollPane1");
506             getJScrollPane1().setViewportView(getScrollPaneTable());
507             // user code begin {1}
508             // user code end
509         } catch (java.lang.Throwable ivjExc) {
510             // user code begin {2}
511             // user code end
512             handleException(ivjExc);
513         }
514     }

```

```

        return ivjJScrollPane1;
516 }
517 /**
518 * Return the JSeparator1 property value.
519 * @return javax.swing.JSeparator
520 */
521 /* WARNING: THIS METHOD WILL BE REGENERATED. */
522 private javax.swing.JSeparator getJSeparator1() {
523     if (ivjJSeparator1 == null) {
524         try {
525             ivjJSeparator1 = new javax.swing.JSeparator();
526             ivjJSeparator1.setName("JSeparator1");
527             // user code begin {1}
528             // user code end
529         } catch (java.lang.Throwable ivjExc) {
530             // user code begin {2}
531             // user code end
532             handleException(ivjExc);
533         }
534     }
535     return ivjJSeparator1;
536 }
537 /**
538 * Return the JToolBar1 property value.
539 * @return javax.swing.JToolBar
540 */
541 /* WARNING: THIS METHOD WILL BE REGENERATED. */
542 private javax.swing.JToolBar getJToolBar1() {
543     if (ivjJToolBar1 == null) {
544         try {
545             ivjJToolBar1 = new javax.swing.JToolBar();
546             ivjJToolBar1.setName("JToolBar1");
547             getJToolBar1().add(getSetupButton(), getSetupButton().getName());
548             ivjJToolBar1.addSeparator();
549             ivjJToolBar1.addSeparator();
550             getJToolBar1().add(getHelpButton(), getHelpButton().getName());
551             getJToolBar1().add(getExitButton(), getExitButton().getName());
552             // user code begin {1}
553             // user code end
554         } catch (java.lang.Throwable ivjExc) {
555             // user code begin {2}
556             // user code end
557             handleException(ivjExc);
558         }
559     }
560     return ivjJToolBar1;
561 }
562 /**
563 * Return the ScrollPaneTable property value.
564 * @return javax.swing.JTable
565 */
566 /* WARNING: THIS METHOD WILL BE REGENERATED. */
567 private javax.swing.JTable getScrollPaneTable() {
568     if (ivjScrollPaneTable == null) {
569         try {
570             ivjScrollPaneTable = new javax.swing.JTable();
571             ivjScrollPaneTable.setName("ScrollPaneTable");
572             getJScrollPane1().setColumnHeaderView(ivjScrollPaneTable.getTableHeader());
573             getJScrollPane1().getViewport().setBackingStoreEnabled(true);
574             ivjScrollPaneTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
575             ivjScrollPaneTable.setPreferredSize(new java.awt.Dimension(716,412));
576             ivjScrollPaneTable.setBounds(0, 0, 716, 412);
577             ivjScrollPaneTable.setPreferredScrollableViewportSize(new java.awt.Dimension(700, 400));
578             ivjScrollPaneTable.setAutoCreateColumnsFromModel(true);
579             // user code begin {1}
580             ivjScrollPaneTable.setModel( getModel() );

```

```

        // 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) {
597     try {
598       ivjServerApplicationJMenuBar = new javax.swing.JMenuBar();
599       ivjServerApplicationJMenuBar.setName("ServerApplicationJMenuBar");
600       ivjServerApplicationJMenuBar.add(getServerMenu());
601       ivjServerApplicationJMenuBar.add(getHelpMenu());
602       // user code begin {1}
603       // user code end
604     } catch (java.lang.Throwable ivjExc) {
605       // user code begin {2}
606       // user code end
607       handleException(ivjExc);
608     }
609   }
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 {
619   if (serverManager == null)
620   {
621     serverManager = new ServerManager();
622     serverManager.setServerApplication( this );
623   }
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) {
633     try {
634       ivjServerMenu = new javax.swing.JMenu();
635       ivjServerMenu.setName("ServerMenu");
636       ivjServerMenu.setText("Setup");
637       ivjServerMenu.add(getSetupMenuItem());
638       ivjServerMenu.add(getJSeparator1());
639       ivjServerMenu.add(getExitMenuItem());
640       // user code begin {1}
641       // user code end
642     } catch (java.lang.Throwable ivjExc) {
643       // user code begin {2}
644       // user code end
645       handleException(ivjExc);
646     }
}

```

```

        }
648     return ivjServerMenu;
}
650 /**
 * Return the ServerSetup property value.
652 * @return serverapp.ServerSetup
 */
654 /* WARNING: THIS METHOD WILL BE REGENERATED. */
655 private ServerSetup getServerSetup() {
656     if (ivjServerSetup == null) {
657         try {
658             ivjServerSetup = new serverapp.ServerSetup();
659             ivjServerSetup.setName("ServerSetup");
660             ivjServerSetup.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
661             // user code begin {1}
662             // user code end
663         } catch (java.lang.Throwable ivjExc) {
664             // user code begin {2}
665             // user code end
666             handleException(ivjExc);
667         }
668     }
669     return ivjServerSetup;
}
670 /**
 * Return the SetupButton property value.
 * @return javax.swing.JButton
 */
674 /*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
676 private javax.swing.JButton getSetupButton() {
677     if (ivjSetupButton == null) {
678         try {
679             ivjSetupButton = new javax.swing.JButton();
680             ivjSetupButton.setName("SetupButton");
681             ivjSetupButton.setToolTipText("Server\u21a6setup");
682             ivjSetupButton.setText("");
683             ivjSetupButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
684             ivjSetupButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
685             ivjSetupButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Properties24.gif")));
686             ivjSetupButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
687             // user code begin {1}
688             // user code end
689         } catch (java.lang.Throwable ivjExc) {
690             // user code begin {2}
691             // user code end
692             handleException(ivjExc);
693         }
694     }
695     return ivjSetupButton;
}
696 /**
 * Return the SetupMenuItem property value.
 * @return javax.swing.JMenuItem
 */
700 /*
 * WARNING: THIS METHOD WILL BE REGENERATED. */
702 private javax.swing.JMenuItem getSetupMenuItem() {
703     if (ivjSetupMenuItem == null) {
704         try {
705             ivjSetupMenuItem = new javax.swing.JMenuItem();
706             ivjSetupMenuItem.setName("SetupMenuItem");
707             ivjSetupMenuItem.setText("Setup");
708             // user code begin {1}
709             // user code end
710         } catch (java.lang.Throwable ivjExc) {
711             // user code begin {2}

```

```

712         // user code end
713         handleException(ivjExc);
714     }
715     return ivjSetupMenuItem;
716 }
717 /**
718 * Insert the method's description here.
719 * Creation date: (17-07-00 21:34:13)
720 * @return javax.swing.table.DefaultTableModel
721 */
722 public javax.swing.table.DefaultTableModel getTableModel() {
723     if ( tableModel == null )
724     {
725         tableModel = new javax.swing.table.DefaultTableModel();
726     }
727     return tableModel;
728 }
729 /**
730 * Called whenever the part throws an exception.
731 * @param exception java.lang.Throwable
732 */
733 private void handleException(java.lang.Throwable exception) {
734
735     /* Uncomment the following lines to print uncaught exceptions to stdout */
736     // System.out.println("----- UNCAUGHT EXCEPTION -----");
737     // exception.printStackTrace(System.out);
738 }
739 /**
740 * Initializes connections
741 * @exception java.lang.Exception The exception description.
742 */
743 /* WARNING: THIS METHOD WILL BE REGENERATED. */
744 private void initConnections() throws java.lang.Exception {
745     // user code begin {1}
746     // user code end
747     getSetupMenuItem().addActionListener(ivjEventHandler);
748     getAboutMenuItem().addActionListener(ivjEventHandler);
749     getExitButton().addActionListener(ivjEventHandler);
750     getHelpButton().addActionListener(ivjEventHandler);
751     getContentsMenuItem().addActionListener(ivjEventHandler);
752     getSetupButton().addActionListener(ivjEventHandler);
753     getExitMenuItem().addActionListener(ivjEventHandler);
754 }
755 /**
756 * Setup group table so that it reflects the current state of the group directory.
757 * Creation date: (17-07-00 17:00:48)
758 */
759 public void initGroupTable()
760 {
761     WorkGroup workGroup = new WorkGroup();
762     tableModel = new javax.swing.table.DefaultTableModel();
763     ivjScrollPaneTable.setModel( tableModel );
764
765     String columnNames[] = WorkGroup.getFieldNames();
766     for ( int i = 0 ; i < columnNames.length; i++ )
767     {
768         tableModel.addColumn( columnNames[i] );
769     }
770     Collection values = serverManager.getServerInfo().getGroups().values();
771     Iterator iterator = values.iterator();
772     while ( iterator.hasNext() )
773     {
774         workGroup = (WorkGroup)iterator.next();
775         Vector v = workGroup.getGroupAsVector();
776         tableModel.addRow( v );

```

```

778     }
780 }
782 /**
783 * Initialize the class.
784 */
784 /* WARNING: THIS METHOD WILL BE REGENERATED. */
785 private void initialize() {
786     try {
787         // user code begin {1}
788         // user code end
789         setName("ServerApplication");
790         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
791         setTitle("PeerView\u00d7server");
792         setSize(598, 528);
793         setJMenuBar(getServerApplicationJMenuBar());
794         setContentPane(getJFrameContentPane());
795         initConnections();
796     } catch (java.lang.Throwable ivjExc) {
797         handleException(ivjExc);
798     }
799     // user code begin {2}
800     getServerManager().setServerApplication( this );
801     getServerManager().configure();
802     center();
803     initTable();
804     updateTable();
805     // user code end
806 }
807 /**
808 * Insert the method's description here.
809 * Creation date: (04-07-00 23:11:34)
810 */
811 public void initTable()
812 {
813     WorkGroup workGroup = new WorkGroup();
814     tableModel = new javax.swing.table.DefaultTableModel();
815     ivjScrollPaneTable.setModel( tableModel );
816
817     String columnNames[] = WorkGroup.getFieldNames();
818     for ( int i = 0 ; i < columnNames.length; i++ )
819     {
820         tableModel.addColumn( columnNames[i] );
821     }
822 }
823 /**
824 * main entrypoint - starts the part when it is run as an application
825 * @param args java.lang.String[]
826 */
827 public static void main(java.lang.String[] args)
828 {
829     parseCommandLineArguments( args );
830     if ( ServerInfo.isVisible() == false )
831     {
832         ServerManager serverManager = new ServerManager();
833         serverManager.configure();
834         return;
835     }
836     try {
837         ServerApplication aServerApplication;
838         aServerApplication = new ServerApplication();
839         aServerApplication.addWindowListener(new java.awt.event.WindowAdapter() {
840             public void windowClosing(java.awt.event.WindowEvent e) {
841                 System.exit(0);
842             };
843         });
844     }

```

```

844     aServerApplication.setVisible( ServerInfo.isVisible() );
845 } catch (Throwable exception) {
846     System.err.println("Exception occurred in main() of javax.swing.JFrame");
847     exception.printStackTrace(System.out);
848 }
849 /**
850 * Parse the array of command line arguments in compliance with the POSIX specifications.
851 * Creation date: (10-09-00 07:54:32)
852 * @param args java.lang.String[]
853 */
854 public static void parseCommandLineArguments(String[] args)
855 {
856     for ( int i = 0 ; i < args.length; i++ )
857     {
858         if ( args[i].equalsIgnoreCase( ServerConstants.getINVISIBLE_ARGUMENT() ) )
859         {
860             ServerInfo.setVisible( false );
861             System.err.println( ServerConstants.getINVISIBLE_MODE_MESSAGE() );
862         }
863         else if ( args[i].equalsIgnoreCase( ServerConstants.getSOCKET_ARGUMENT() ) )
864         {
865             ServerInfo.getPreferences().put( ServerConstants.getConnection_TYPE(), ServerConstants.getSOCKET()
866                                             () );
867         }
868         else if ( args[i].equalsIgnoreCase( ServerConstants.getHTTP_ARGUMENT() ) )
869         {
870             ServerInfo.getPreferences().put( ServerConstants.getConnection_TYPE(), ServerConstants.getHTTP()
871                                             () );
872         }
873     }
874 /**
875 * Insert the method's description here.
876 * Creation date: (29-06-00 18:47:13)
877 * @param newServerManager serverapp.ServerManager
878 */
879 public void setServerManager(ServerManager newServerManager) {
880     serverManager = newServerManager;
881 }
882 /**
883 * Insert the method's description here.
884 * Creation date: (17-07-00 21:34:13)
885 * @param newTableModel javax.swing.table.DefaultTableModel
886 */
887 public void setTableModel(javax.swing.table.DefaultTableModel newTableModel) {
888     tableModel = newTableModel;
889 }
890 /**
891 * Insert the method's description here.
892 * Creation date: (17-08-00 00:22:47)
893 */
894 public void terminateAndClose()
895 {
896     getServerManager().terminate();
897     dispose();
898 }
899 /**
900 * Update the table underlying the server copy of the group directory.
901 * Creation date: (22-07-00 23:35:40)
902 */
903 public void updateTable()
904 {
905     // clear the table
906     for ( int i = 0; i < tableModel.getRowCount(); i++ )
907     {

```

```

908         tableModel.removeRow( 0 );
909     }
910     tableModel.setNumRows( 0 );
911     // update with contents of group directory
912     Collection values = getServerManager().getServerInfo().getGroups().values();
913     Iterator iterator = values.iterator();
914     while ( iterator.hasNext() )
915     {
916         WorkGroup workGroup = (WorkGroup) iterator.next();
917         Vector v = workGroup.getGroupAsVector();
918         tableModel.addRow( v );
919     }
920     // Distribute surplus space evenly
921     getScrollPaneTable().sizeColumnsToFit( -1 );
922     getScrollPaneTable().revalidate();
923     getScrollPaneTable().repaint();
924     getJScrollPane1().revalidate();
925     getJScrollPane1().repaint();
926 }

```

Listing C.30: ServerConstants.java

```

package serverapp;
2
4 import javax.swing.border.*;
4 import javax.swing.BorderFactory.*;
5 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 *
11 * Creation date: (27-06-00 20:58:01)
12 * @author:
13 */
14
15 public class ServerConstants extends SharedConstants
16 {
17     private final static java.lang.String PREFERENCES_FILE_NAME = "sprefs.ini";
18     private final static java.lang.String GROUPS_FILE_NAME = "groups.dat";
19     private final static java.lang.String GROUP_TABLE_CAST_FAILURE_MESSAGE = "Could\u00b9not\u00b9complete\u00b9read\u00b9of\u00b9
        group\u00b9table\u00b9from\u00b9disk";
20     private final static java.lang.String ROOT_NODE_MESSAGE_TITLE = "System\u00b9notice";
21     private final static java.lang.String PROPERTIES_HEADER = "";
22     private final static java.lang.String FRAME_HORIZONTAL_SIZE = "Frame\u00b9horizontal\u00b9size";
23     private final static java.lang.String FRAME_VERTICAL_SIZE = "Frame\u00b9vertical\u00b9size";
24     private final static java.lang.String DEFAULT_FRAME_HORIZONTAL_SIZE = "0";
25     private final static java.lang.String DEFAULT_FRAME_VERTICAL_SIZE = "0";
26     private final static java.lang.String FRAME_X_COORDINATE = "Frame\u00b9x\u00b9coordinate";
27     private final static java.lang.String FRAME_Y_COORDINATE = "Frame\u00b9y\u00b9coordinate";
28     private final static java.lang.String DEFAULT_FRAME_X_COORDINATE = "0";
29     private final static java.lang.String DEFAULT_FRAME_Y_COORDINATE = "0";
30     final static String DEFAULT_PREFERENCES[][] = { { SERVER_NAME, DEFAULT_SERVER_NAME },
31                                                 { SERVER_PORT, String.valueOf(DEFAULT_SERVER_PORT) },
32                                                 { CONNECTION_TYPE, DEFAULT_CONNECTION_TYPE },
33                                                 { FRAME_HORIZONTAL_SIZE, DEFAULT_FRAME_HORIZONTAL_SIZE },
34                                                 { FRAME_VERTICAL_SIZE, DEFAULT_FRAME_VERTICAL_SIZE },
35                                                 { FRAME_X_COORDINATE, DEFAULT_FRAME_X_COORDINATE },
36                                                 { FRAME_Y_COORDINATE, DEFAULT_FRAME_Y_COORDINATE },
37                                                 { getREGISTRY_PORT(), getDefaultRegistryPort() },
38 };
40     private final static double FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE = 0.5;
41     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  

43         from disk.";
44     private final static java.lang.String COULD_NOT_WRITE_MESSAGES = "Could not write the message table  

45         to disk.";
46     private final static java.lang.String COULD_NOT_READ_GROUPS = "Could not read the group directory  

47         from disk.";
48     private final static java.lang.String COULD_NOT_WRITE_GROUPS = "Could not write the group directory  

49         to disk.";
50     private final static java.lang.String INVISIBLE_ARGUMENT = "-invisible";
51     private final static java.lang.String INVISIBLE_MODE_MESSAGE = "Running in invisible mode - the GUI  

52         will be disabled.";
53     private final static java.lang.String COULD_NOT_CLOSE_GROUP_FILE = "Could not close group directory  

54         file. The directory may not have been properly initialized.";
55     /**
56      * ServerConstants constructor comment.
57      */
58     public ServerConstants() {
59         super();
60     }
61     /**
62      * Insert the method's description here.
63      * Creation date: (05-10-00 09:48:05)
64      * @return java.lang.String
65      */
66     public final static java.lang.String getCOULD_NOT_CLOSE_GROUP_FILE() {
67         return COULD_NOT_CLOSE_GROUP_FILE;
68     }
69     /**
70      * Insert the method's description here.
71      * Creation date: (15-08-00 08:41:01)
72      * @return java.lang.String
73      */
74     public final static java.lang.String getCOULD_NOT_READ_GROUPS() {
75         return COULD_NOT_READ_GROUPS;
76     }
77     /**
78      * Insert the method's description here.
79      * Creation date: (14-08-00 23:24:44)
80      * @return java.lang.String
81      */
82     public final static java.lang.String getCOULD_NOT_READ_MESSAGES() {
83         return COULD_NOT_READ_MESSAGES;
84     }
85     /**
86      * Insert the method's description here.
87      * Creation date: (15-08-00 08:42:08)
88      * @return java.lang.String
89      */
90     public final static java.lang.String getCOULD_NOT_WRITE_GROUPS() {
91         return COULD_NOT_WRITE_GROUPS;
92     }
93     /**
94      * Insert the method's description here.
95      * Creation date: (14-08-00 23:29:55)
96      * @return java.lang.String
97      */
98     public final static java.lang.String getCOULD_NOT_WRITE_MESSAGES() {
99         return COULD_NOT_WRITE_MESSAGES;
100    }
101    /**
102       * Insert the method's description here.
103       * Creation date: (05-08-00 19:49:43)
104       * @return java.lang.String
105       */
106    public final static java.lang.String getDEFAULT_FRAME_HORIZONTAL_SIZE() {
107        return DEFAULT_FRAME_HORIZONTAL_SIZE;

```

```

102 }
103 /**
104 * Insert the method's description here.
105 * Creation date: (05-08-00 19:50:26)
106 * @return java.lang.String
107 */
108 public final static java.lang.String getDEFAULT_FRAME_VERTICAL_SIZE() {
109     return DEFAULT_FRAME_VERTICAL_SIZE;
110 }
111 /**
112 * Insert the method's description here.
113 * Creation date: (05-08-00 19:51:49)
114 * @return java.lang.String
115 */
116 public final static java.lang.String getDEFAULT_FRAME_X_COORDINATE() {
117     return DEFAULT_FRAME_X_COORDINATE;
118 }
119 /**
120 * Insert the method's description here.
121 * Creation date: (05-08-00 19:52:26)
122 * @return java.lang.String
123 */
124 public final static java.lang.String getDEFAULT_FRAME_Y_COORDINATE() {
125     return DEFAULT_FRAME_Y_COORDINATE;
126 }
127 /**
128 * Insert the method's description here.
129 * Creation date: (03-07-00 14:34:27)
130 * @return java.util.Properties
131 */
132 public static java.util.Properties getDEFAULT_PREFERENCES()
133 {
134     Properties p = new Properties();
135
136     for (int i = 0; i < DEFAULT_PREFERENCES.length; i++)
137     {
138         p.setProperty( DEFAULT_PREFERENCES[i][0], DEFAULT_PREFERENCES[i][1] );
139     }
140     return p;
141 }
142 /**
143 * Insert the method's description here.
144 * Creation date: (05-08-00 19:49:06)
145 * @return java.lang.String
146 */
147 public final static java.lang.String getFRAME_HORIZONTAL_SIZE() {
148     return FRAME_HORIZONTAL_SIZE;
149 }
150 /**
151 * Insert the method's description here.
152 * Creation date: (05-08-00 20:05:17)
153 * @return int
154 */
155 public final static double getFRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE() {
156     return FRAME_SIZE_AS_FRACTION_OF_SCREEN_SIZE;
157 }
158 /**
159 * Insert the method's description here.
160 * Creation date: (05-08-00 19:49:24)
161 * @return java.lang.String
162 */
163 public final static java.lang.String getFRAME_VERTICAL_SIZE() {
164     return FRAME_VERTICAL_SIZE;
165 }
166 /**
167 * Insert the method's description here.

```

```

168 * Creation date: (05-08-00 19:51:18)
169 * @return java.lang.String
170 */
171 public final static java.lang.String getFRAME_X_COORDINATE() {
172     return FRAME_X_COORDINATE;
173 }
174 /**
175 * Insert the method's description here.
176 * Creation date: (05-08-00 19:51:34)
177 * @return java.lang.String
178 */
179 public final static java.lang.String getFRAME_Y_COORDINATE() {
180     return FRAME_Y_COORDINATE;
181 }
182 /**
183 * Insert the method's description here.
184 * Creation date: (04-07-00 15:57:55)
185 * @return java.lang.String
186 */
187 public final static java.lang.String getGROUP_TABLE_CAST_FAILURE_MESSAGE() {
188     return GROUP_TABLE_CAST_FAILURE_MESSAGE;
189 }
190 /**
191 * Insert the method's description here.
192 * Creation date: (04-07-00 15:19:40)
193 * @return java.lang.String
194 */
195 public final static java.lang.String getGROUPS_FILE_NAME() {
196     return GROUPS_FILE_NAME;
197 }
198 /**
199 * Insert the method's description here.
200 * Creation date: (10-09-00 07:52:35)
201 * @return java.lang.String
202 */
203 public final static java.lang.String getINVISIBLE_ARGUMENT() {
204     return INVISIBLE_ARGUMENT;
205 }
206 /**
207 * Insert the method's description here.
208 * Creation date: (10-09-00 08:08:58)
209 * @return java.lang.String
210 */
211 public final static java.lang.String getINVISIBLE_MODE_MESSAGE() {
212     return INVISIBLE_MODE_MESSAGE;
213 }
214 /**
215 * Insert the method's description here.
216 * Creation date: (14-08-00 23:19:47)
217 * @return java.lang.String
218 */
219 public final static java.lang.String getMESSAGES_FILE_NAME() {
220     return MESSAGES_FILE_NAME;
221 }
222 /**
223 * Insert the method's description here.
224 * Creation date: (03-07-00 14:43:22)
225 * @return java.lang.String
226 */
227 public final static java.lang.String getPREFERENCES_FILE_NAME() {
228     return PREFERENCES_FILE_NAME;
229 }
230 /**
231 * Insert the method's description here.
232 * Creation date: (05-08-00 19:44:31)
233 * @return java.lang.String

```

```

234  /*
235   * Insert the method's description here.
236   * Creation date: (10-07-00 00:50:05)
237   * @param documentName java.lang.String
238   * @param senderName java.lang.String
239   */
240
241     public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
242           Date creationDate)
243 {
244     return "This is the discussion forum for the document " + documentName + " which was added by " +
245           senderName +
246           " on " + creationDate.toString();
247 }
248 /**
249  * Insert the method's description here.
250  * Creation date: (10-07-00 00:50:05)
251  * @param documentName java.lang.String
252  * @param senderName java.lang.String
253  */
254
255     public static String getROOT_NODE_MESSAGE_CONTENTS(String documentName, String senderName, java.util.
256           GregorianCalendar creationDate)
257 {
258     return "This is the discussion forum for the document " + documentName + " which was added by " +
259           senderName +
260           " on " + creationDate.getTime().toString();
261 }
262 /**
263  * Insert the method's description here.
264  * Creation date: (10-07-00 00:48:37)
265  * @return java.lang.String
266  */
267     public final static java.lang.String getROOT_NODE_MESSAGE_TITLE() {
268     return ROOT_NODE_MESSAGE_TITLE;
269   }
270 }
```

Listing C.31: ServerInfo.java

```

package serverapp;
2
3   import java.util.Properties;
4   import java.util.Hashtable;
5   import peerviewmisc.*;
6   import java.io.*;
7
8 /**
9  * The PeerView server main entity object. This class holds data structures and fields shared by several
10 * server components.
11 * Creation date: (28-06-00 22:16:34)
12 * @author:
13 */
14
15 public class ServerInfo
16 {
17   private com.sun.media.jsdt.Channel groupManagementChannel = null;
18   private java.util.Hashtable groups = new Hashtable();
19   private static java.util.Properties preferences = null;
20   private com.sun.media.jsdt.Session groupManagementSession;
21   private int mostRecentPortNumber = ServerConstants.getDEFAULT_SERVER_PORT();
22   private java.util.Hashtable discussions = new Hashtable();
23   private java.util.Hashtable messages = new Hashtable();
```

```

    private static boolean visible = true;
24 /**
25  * ServerInfo constructor comment.
26 */
27 public ServerInfo()
28 {
29     super();
30 }
31 /**
32  * Insert the method's description here.
33  * Creation date: (14-08-00 23:05:59)
34  * @exception java.lang.Throwable The exception description.
35 */
36 protected void finalize() throws java.lang.Throwable
37 {
38 }
39 /**
40  * Insert the method's description here.
41  * Creation date: (09-07-00 15:03:03)
42  * @return java.util.Hashtable
43 */
44 public java.util.Hashtable getDiscussions() {
45     return discussions;
46 }
47 /**
48  * Insert the method's description here.
49  * Creation date: (01-07-00 01:20:55)
50  * @return com.sun.media.jsdt.Channel
51 */
52 public com.sun.media.jsdt.Channel getGroupManagementChannel() {
53     return groupManagementChannel;
54 }
55 /**
56  * Insert the method's description here.
57  * Creation date: (04-07-00 17:12:12)
58  * @return com.sun.media.jsdt.Session
59 */
60 public com.sun.media.jsdt.Session getGroupManagementSession() {
61     return groupManagementSession;
62 }
63 /**
64  * Insert the method's description here.
65  * Creation date: (03-07-00 12:11:40)
66  * @return java.util.Properties
67 */
68 public java.util.Hashtable getGroups()
69 {
70     if ( groups == null )
71     {
72         groups = new Hashtable();
73     }
74     return groups;
75 }
76 /**
77  * Insert the method's description here.
78  * Creation date: (10-07-00 01:27:49)
79  * @return java.util.Hashtable
80 */
81 public java.util.Hashtable getMessages()
82 {
83     return messages;
84 }
85 /**
86  * Insert the method's description here.
87  * Creation date: (07-07-00 17:30:38)
88  * @return int
89 */

```

```

    public int getMostRecentPortNumber() {
90        return mostRecentPortNumber;
    }
92 /**
 * Insert the method's description here.
94 * Creation date: (02-07-00 22:08:51)
 * @return java.util.Properties
96 */
97 public static java.util.Properties getPreferences()
98 {
99     if ( preferences == null)
100    {
101        preferences = new Properties( ServerConstants.getDEFAULT_PREFERENCES() );
102        readPreferences();
103    }
104    return preferences;
}
105 /**
 * Insert the method's description here.
106 * Creation date: (10-09-00 07:59:29)
 * @return boolean
107 */
108 public static boolean isVisible() {
109     return visible;
}
110 /**
 * Read preferences from Properties object on disk. Each call to this function results in a read
111 * 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 negligible as the file in
 * question is essentially
112 * plain text.
 * Algorithm:
113 * 1. Construct new Properties object P
 * 2. Initialize P with default preferences
114 * 3. Read user preferences from disk file
 * 4. IF read is successful THEN
115 * 4.1. Store user preferences in P
 * 5. Return P
116 * This approach will return a valid object even if the read fails or in case of the application being
 * used for the first
 * time.
117 * Creation date: (25-06-00 13:54:55)
 */
118 public static void readPreferences()
{
119     try
120     {
121         FileInputStream f = new FileInputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
122         getPreferences().load(f);
123         f.close();
124     }
125     catch (FileNotFoundException ffe)
126     {
127         // the exception can be handled by ignoring it since the default values in p will substitute for
 * what should
 * have been read from file.
128     }
129     catch (IOException ie)
130     {
131         // any IOException != FileNotFoundException should be dealt with in the conventional manner
132         System.err.println( ie.toString() );
133         System.exit(0);
134     }
}
135 /**

```

```

152 * Insert the method's description here.
153 * Creation date: (09-07-00 15:03:03)
154 * @param newDiscussions java.util.Hashtable
155 */
156 public void setDiscussions(java.util.Hashtable newDiscussions) {
157     discussions = newDiscussions;
158 }
159 /**
160 * Insert the method's description here.
161 * Creation date: (01-07-00 01:20:55)
162 * @param newGroupManagementChannel com.sun.media.jsdt.Channel
163 */
164 public void setGroupManagementChannel(com.sun.media.jsdt.Channel newGroupManagementChannel) {
165     groupManagementChannel = newGroupManagementChannel;
166 }
167 /**
168 * Insert the method's description here.
169 * Creation date: (04-07-00 17:12:12)
170 * @param newGroupManagementSession com.sun.media.jsdt.Session
171 */
172 public void setGroupManagementSession(com.sun.media.jsdt.Session newGroupManagementSession) {
173     groupManagementSession = newGroupManagementSession;
174 }
175 /**
176 * Insert the method's description here.
177 * Creation date: (03-07-00 12:11:40)
178 * @param newGroups java.util.Properties
179 */
180 public void setGroups(java.util.Hashtable newGroups) {
181     groups = newGroups;
182 }
183 /**
184 * Insert the method's description here.
185 * Creation date: (10-07-00 01:27:49)
186 * @param newMessages java.util.Hashtable
187 */
188 public void setMessages(java.util.Hashtable newMessages) {
189     messages = newMessages;
190 }
191 /**
192 * Insert the method's description here.
193 * Creation date: (07-07-00 17:30:38)
194 * @param newMostRecentPortNumber int
195 */
196 public void setMostRecentPortNumber(int newMostRecentPortNumber) {
197     mostRecentPortNumber = newMostRecentPortNumber;
198 }
199 /**
200 * Insert the method's description here.
201 * Creation date: (03-07-00 14:38:45)
202 * @param newPreferences java.util.Properties
203 */
204 public void setPreferences(java.util.Properties newPreferences) {
205     preferences = newPreferences;
206 }
207 /**
208 * Insert the method's description here.
209 * Creation date: (10-09-00 07:59:29)
210 * @param newVisible boolean
211 */
212 public static void setVisible(boolean newVisible) {
213     visible = newVisible;
214 }
215 /**
216 * Write preferences to disk file.
217 * Creation date: (17-07-00 12:13:32)

```

```

218 * @param preferences java.util.Properties
219 */
220 public static void writePreferences()
221 {
222     // write Properties object to disk
223     try
224     {
225         FileOutputStream f = new FileOutputStream( ServerConstants.getPREFERENCES_FILE_NAME() );
226         getPreferences().store( f, ServerConstants.getPropertyHeader() );
227         f.close();
228     }
229     catch (IOException ie)
230     {
231         System.err.println( ie.toString() );
232         System.exit(0);
233     }
234 }

```

Listing C.32: ServerManager.java

```

package serverapp;
2
3     import com.sun.media.jsdt.*;
4
5     import java.io.*;
6     import java.util.Hashtable;
7
8     import peerviewmisc.*;
9     import java.util.Collection;
10
11    import java.util.Iterator;
12    import javax.swing.tree.*;
13
14    import java.util.Date;
15
16    import com.sun.media.jsdt.event.*;
17
18 /**
19  * This is the main control class for the PeerView server application. It contains methods and control
20  * structures shared
21  * by several server components.
22  * Creation date: (27-06-00 20:17:35)
23  * @author:
24  */
25
26 public class ServerManager
27 {
28     private ServerInfo serverInfo = null;
29
30     public class DocNameAndSize implements Serializable
31     {
32         public String docName = null;
33         public long size = 0;
34
35         public DocNameAndSize( String docNameArg, long sizeArg )
36         {
37             docName = docNameArg;
38             size = sizeArg;
39         }
40
41         public String getDocName()
42         {
43             return docName;
44         }
45
46         public void setDocName( String newDocName )
47         {
48             docName = newDocName;
49         }
50
51         public long getSize()
52         {
53             return size;
54         }
55     }
56
57 }

```

```

        public void setSize( long newSize )
46    {
48        size = newSize;
49    }
50};

51 public class GroupListener extends SessionAdaptor
52 {
53     public String workGroupID = null;
54     public GroupListener( String workGroupIDArg )
55     {
56         workGroupID = workGroupIDArg;
57     }
58
59     public void sessionJoined( SessionEvent se )
60     {
61         clientJoinedGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID
62             ) );
63     }
64
65     public void sessionLeft( SessionEvent se )
66     {
67         System.out.println( "ExitClient:" + se.getClientName() );
68         try
69         {
70             for ( int i = 0 ; i < se.getSession().listClientNames().length; i++ )
71             {
72                 System.out.println( se.getSession().listClientNames()[i] );
73             }
74         }
75         catch ( Exception E )
76         {
77             clientLeftGroup( se.getClientName(), (WorkGroup) getServerInfo().getGroups().get( workGroupID ) )
78             ;
79         }
80     }
81 }

82 public class DefaultClient implements com.sun.media.jsdt.Client
83 {
84     String name;
85
86     public DefaultClient (String nameArg )
87     {
88         name = nameArg;
89     }
90     public Object authenticate ( com.sun.media.jsdt.AuthenticationInfo ai )
91     {
92         return null;
93     }
94     public String getName ()
95     {
96         return name;
97     }
98 };
99

100 public class GroupManagementServer implements Client, ChannelConsumer
101 {
102     String name;
103
104     public GroupManagementServer()
105     {
106
107         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\f" + 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() ).
167             equalsIgnoreCase( ServerConstants.getTOPIC_TREE_UPDATE() ) )
168     {
169         updateTopicTree( (String) incomingPackage.getProperties().getProperty( DataPackage.getDOCPATH
170                         () ), (DefaultTreeModel) incomingPackage.getContents() );
171     }
172
173     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
174             equalsIgnoreCase( ServerConstants.getDELETE_MESSAGE() ) )
175     {
176         deleteMessage( (String) incomingPackage.getContents() );
177     }
178     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
179             equalsIgnoreCase( ServerConstants.getWORKGROUP_UPDATE_FROM_CLIENT() ) )
180     {
181         updateWorkGroup( (WorkGroup) incomingPackage.getContents() );
182     }
183     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
184             equalsIgnoreCase( ServerConstants.getREQUEST_REMOVE_GROUP() ) )
185     {
186         removeWorkGroup( (WorkGroup) incomingPackage.getContents() );
187     }
188     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
189             equalsIgnoreCase( ServerConstants.getNEW_DOCUMENTS() ) )
190     {
191         clientAddedDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty(
192                         DataPackage.getAUTHOR_NAME() ), incomingPackage.getProperties().getProperty( DataPackage
193                             .getWORKGROUP_ID() ), (java.util.Vector) incomingPackage.getContents() );
194         broadcastGroupDirectory();
195     }
196     else if ( incomingPackage.getProperties().getProperty( DataPackage.getDESCRIPTION() ).
197             equalsIgnoreCase( ServerConstants.getREMOVE_DOCUMENTS() ) )
198     {
199         clientRemovedDocuments( data.getSenderName(), incomingPackage.getProperties().getProperty(
200                         DataPackage.getWORKGROUP_ID() ), (java.util.Vector) incomingPackage.getContents() );
201         broadcastGroupDirectory();
202     }
203
204     private PeerViewServer defaultServer = new PeerViewServer(ServerConstants.getDEFAULT_SERVER_NAME());
205     private GroupManagementServer groupManagementServer = null;
206     private ServerApplication serverApplication = null;
207 /**
208 * ServerManager constructor comment.
209 */
210 public ServerManager() {
211     super();
212     initialize();
213 }
214 /**
215 * .
216 * Creation date: (07-07-00 02:16:02)
217 * @param group peerviewmisc.WorkGroup

```

```

    */
220 public void activateGroup(WorkGroup workGroup)
{
222     Session session = null;
223     urlString url = null;
224     GroupManagementServer newServer = new GroupManagementServer();
225     newServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_name() ) );
226
227     try
228     {
229         boolean created = false;
230     /* // Cycle through port numbers until an unoccupied one is found and use it for creating a new
231         session
232         while ( created == false )
233         {
234             try
235             {
236                 url = urlString.createSessionURL( ServerInfo.getPreferences().getProperty ( ServerConstants.
237                     getServer_name() ),
238                     Integer.parseInt( ServerInfo.getPreferences().getProperty(
239                         ServerConstants.getServer_port() ),
240                         ServerInfo.getPreferences().getProperty ( ServerConstants.
241                             getConnection_type() ),
242                         workGroup.getWorkGroupID() );
243                 // System.out.println( workGroup.getWorkGroupID() );
244                 session = SessionFactory.createSession( newServer, url, true );
245                 created = true;
246                 session.addSessionListener( new GroupListener( workGroup.getWorkGroupID() ) );
247             }
248             catch ( PortInUseException PIUE )
249             {
250                 serverInfo.setMostRecentPortNumber( serverInfo.getMostRecentPortNumber() + 1 );
251             }
252         }
253     */
254     Channel groupChannel = session.createChannel( newServer, workGroup.getWorkGroupChannelID(), true,
255         true, true );
256     groupChannel.addConsumer( newServer, newServer );
257     workGroup.setSessionURL( url );
258
259     catch ( JSDTEException JSDTE )
260     {
261         handleJSDTEException( JSDTE );
262     }
263
264 }
265 /**
266 * Activate the groups read from disk file by creating and registering the appropriate sessions and
267 * channels.
268 * Creation date: (05-07-00 23:33:46)
269 */
270 public void activateGroups()
271 {
272     Collection values = serverInfo.getGroups().values();
273
274     Iterator iterator = values.iterator();
275     while ( iterator.hasNext() )
276     {
277         activateGroup( (WorkGroup) iterator.next() );
278     }
279 }
280 /**
281 * Insert the method's description here.
282 * Creation date: (03-07-00 19:05:53)
283 * @param workGroup peerviewmisc.WorkGroup

```

```

    */
280 public void addGroup(peerviewmisc.WorkGroup workGroup)
{
282     serverInfo.getGroups().put( workGroup.getWorkGroupID(), workGroup);
283     activateGroup( workGroup );
284     broadcastGroupDirectory();
285     serverApplication.updateTable();
286 }
/***
288 * Insert the method's description here.
289 * Creation date: (11-07-00 21:12:23)
290 * @param peerviewmisc.Message
291 */
292 public void addMessage(Message newMessage)
{
294     getServerInfo().getMessages().put( newMessage.getMessageID(), newMessage );
295 }
296 /**
297 * Insert the method's description here.
298 * Creation date: (20-09-00 03:03:54)
299 * @param senderName java.lang.String
300 * @param documents java.util.Vector
301 */
302 public void broadcastDocumentRemovals(String senderName, WorkGroup workGroup, java.util.Vector documents
)
{
304     DataPackage outgoingPackage = new DataPackage();
305     outgoingPackage.setContents( documents );
306     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
307         getREMOVE_DOCUMENTS() );
308     outgoingPackage.getProperties().setProperty( DataPackage.getWORKGROUP_ID(), workGroup.getWorkGroupID
()
);
309     outgoingPackage.getProperties().setProperty( DataPackage.getCLIENT_NAME(), senderName );
310     outgoingPackage.getProperties().setProperty( DataPackage.getAUTHOR_NAME(), ServerInfo.getPreferences
()
.getProperty( ServerConstants.getServerNAME() ) );
311     try
312     {
313         urlString url = workGroup.getSessionURL();
314         Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
315         Channel groupChannel = session.createChannel( getGroupManagementServer(), workGroup.
316             getWorkGroupChannelID(), true, true, false );
317         Data data = new Data( outgoingPackage );
318         groupChannel.sendToOthers( getGroupManagementServer(), data );
319     }
320     catch ( JSDTEException JSDTE )
321     {
322         handleJSDTEException( JSDTE );
323     }
324 }
/***
325 * Insert the method's description here.
326 * Creation date: (22-07-00 20:08:48)
327 * @param workGroup peerviewmisc.WorkGroup
328 */
329 public void broadcastGroupDirectory()
{
330     DataPackage outgoingPackage = new DataPackage();

332     outgoingPackage.setContents( getServerInfo().getGroups() );
333     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
334         getGROUP_DIRECTORY() );
335     try
336     {
337         if ( getServerInfo().getGroupManagementChannel() != null )
338         {
339             getServerInfo().getGroupManagementChannel().sendToOthers( getGroupManagementServer(),

```

```

340         }
341     }
342     catch ( JSDTEException JSDTE )
343     {
344         handleJSDTEException( JSDTE );
345     }
346     if ( getServerApplication() != null )
347     {
348         getServerApplication().updateTable();
349     }
350 }
351 /**
352 * Insert the method's description here.
353 * Creation date: (27-08-00 20:58:01)
354 */
355 public void broadcastTopicTree( String documentName, DefaultTreeModel treeUpdate)
356 {
357     DataPackage outgoingPackage = new DataPackage();
358     outgoingPackage.setContents( treeUpdate );
359     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
360         getTOPIC_TREE_UPDATE() );
361     outgoingPackage.getProperties().setProperty( DataPackage.getDOCSPATH(), documentName );
362     try
363     {
364         getServerInfo().getGroupManagementChannel().sendToOthers( getGroupManagementServer(),
365             new Data( outgoingPackage ) );
366     }
367     catch ( JSDTEException JSDTE )
368     {
369         handleJSDTEException( JSDTE );
370     }
371 }
372 /**
373 * Insert the method's description here.
374 * Creation date: (19-09-00 18:40:28)
375 * @param senderName java.lang.String
376 * @param groupID java.lang.String
377 * @param documents java.util.Vector
378 */
379 public void clientAddedDocuments(String senderName, String authorName, String groupID, java.util.Vector
380 documents)
381 {
382     WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupID );
383     for ( int i = 0 ; i < documents.size(); i++ )
384     {
385         CompressedDocumentPackage docPackage = (CompressedDocumentPackage) documents.get( i );
386         Pair docNameAndSize = new Pair( docPackage.getPath(), String.valueOf( docPackage.
387             getUncompressedLength() ) );
388         ( (java.util.Vector) groupToBeUpdated.getDocuments().get( senderName )).add( docNameAndSize );
389         groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + docPackage.
390             getUncompressedLength() );
391         groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() + 1 );
392     }
393     getServerInfo().getGroups().put( groupID, groupToBeUpdated );
394 }
395 /**
396 * Update the data structure holding the groups and submit it to all clients connected to the group
397 management channel.
398 * Creation date: (19-09-00 17:55:14)
399 * @param clientName java.lang.String
400 * @param workGroup peerviewmisc.WorkGroup

```

```

400  */
401  public void clientJoinedGroup(String clientName, WorkGroup workGroup)
402  {
403      WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID() );
404
405      groupToBeUpdated.getParticipants().add( clientName );
406      groupToBeUpdated.getDocuments().put( clientName, new java.util.Vector() );
407      getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
408      broadcastGroupDirectory();
409  }
410 /**
411  * Forcibly expel client from group management session if it has not already left. This to ensure
412  * consistency when clients
413  * terminate abruptly, i.e. do not exit properly.
414  * Creation date: (01-11-00 23:45:19)
415  * @param clientName java.lang.String
416  */
417  public void clientLeft(String clientName)
418  {
419      try
420      {
421          for ( int i = 0 ; i < getServerInfo().getGroupManagementSession().listClientNames().length; i++ )
422          {
423              if ( getServerInfo().getGroupManagementSession().listClientNames()[i].equalsIgnoreCase(
424                  clientName ) )
425              {
426                  //System.out.println( "Found client : " + clientName );
427                  DefaultClient tempClient = new DefaultClient( clientName );
428                  getServerInfo().getGroupManagementSession().leave( tempClient );
429                  break;
430              }
431          }
432      }
433      catch ( JSDTEException JSDTE )
434      {
435          handleJSDTEException( JSDTE );
436      }
437  }
438 /**
439  * This method is triggered when a client leaves a group. The server updates the group's entry in the
440  * group directory
441  * and resubmits it to other clients. Client that partake in the group that the client left are notified
442  * of which
443  * documents to remove from their panoramas.
444  * Creation date: (19-09-00 17:55:46)
445  * @param clientName java.lang.String
446  * @param workGroup peerviewmisc.WorkGroup
447  */
448  public void clientLeftGroup(String clientName, WorkGroup workGroup)
449  {
450      WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID() );
451
452      if ( groupToBeUpdated.getParticipants().contains( clientName ) )
453      {
454          java.util.Vector documentNames = new java.util.Vector();
455          java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( clientName );
456          Iterator docIterator = documents.iterator();
457          while ( docIterator.hasNext() )
458          {
459              Pair docNameAndSize = (Pair) docIterator.next();
460              groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt(
461                  docNameAndSize.getValue() ) );
462              groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
463              documentNames.add( docNameAndSize.getKey() );
464

```

```

        }
460    broadcastDocumentRemovals( clientName, groupToBeUpdated, documentNames );
462
463    groupToBeUpdated.getParticipants().remove( clientName );
464    groupToBeUpdated.getDocuments().remove( clientName );
465    getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
466    broadcastGroupDirectory();
467}
468 /**
469 * This method is triggered when a client removes one or more documents from his or her panorama. The
470 * server updates the
471 * group that the client belongs to and resubmits the group directory to all clients.
472 * Creation date: (19-09-00 19:08:00)
473 * @param files java.util.HashSet
474 */
475 public void clientRemovedDocuments(String senderName, String groupID, java.util.Vector filePaths)
476 {
477     WorkGroup groupToBeUpdated = (WorkGroup) getServerInfo().getGroups().get( groupID );
478
479     Iterator iterator = filePaths.iterator();
480     while ( iterator.hasNext() )
481     {
482         String path = (String) iterator.next();
483         java.util.Vector documents = (java.util.Vector) groupToBeUpdated.getDocuments().get( senderName );
484         Iterator docIterator = documents.iterator();
485         Pair docNameAndSize = null;
486         while ( docIterator.hasNext() )
487         {
488             docNameAndSize = (Pair) docIterator.next();
489             if ( docNameAndSize.getKey().equals( path ) )
490             {
491                 groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() - Integer.parseInt
492                     ( docNameAndSize.getValue() ) );
493                 groupToBeUpdated.setNumberOfDocuments( groupToBeUpdated.getNumberOfDocuments() - 1 );
494                 documents.remove( docNameAndSize );
495                 break;
496             }
497         }
498     }
499     getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
500 }
501 /**
502 * Perform miscellaneous initialization operations.
503 * Creation date: (10-10-00 23:12:57)
504 */
505 public void configure()
506 {
507     startRegistry();
508     createGroupManagement();
509     readMessages();
510     readGroups();
511     readDiscussions();
512 }
513 /**
514 * Insert the method's description here.
515 * Creation date: (29-06-00 08:57:23)
516 * @return java.lang.String
517 * @param sessionName java.lang.String
518 */
519 public String createClientName(String sessionName)
520 {
521     return ServerInfo.getProperty( ServerConstants.getServerName() )+ sessionName;
522 }
523 /**

```

```

522 * Create a new discussion forum and initialize it to the default structure, namely a root node and
523 * system notification
524 * Creation date: (09-07-00 14:55:18)
525 * @param newDocument java.io.File
526 */
527 public void createDiscussionForum(String authorName, String newDocument, String senderName)
528 {
529     String fqDocumentName = ServerConstants.createFullyQualifiedDocumentName( senderName, newDocument );
530     // if a discussion forum for this document does not already exist then create one
531     if ( getServerInfo().getDiscussions().containsKey( fqDocumentName )== false )
532     {
533         Message message = new Message();
534         message.setAuthor( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ) );
535         message.setCreationDate( new java.util.Date() );
536         message.setCreator( ServerInfo.getPreferences().getProperty( ServerConstants.getServer_NAME() ) );
537         message.setTitle( ServerConstants.getROOT_NODE_MESSAGE_TITLE() );
538         message.setContents( ServerConstants.getROOT_NODE_MESSAGE_CONTENTS( newDocument, authorName,
539             message.getCreationDate() ) );
540         message.setDocument( newDocument );
541         getServerInfo().getMessages().put( message.getMessageID(), message );
542         // udskift med konstant
543         DefaultMutableTreeNode innerNode = new DefaultMutableTreeNode( newDocument + ",authored_by" +
544             authorName, true );
545         DefaultMutableTreeNode treeNode = new DefaultMutableTreeNode( message.getMessageID(), false );
546         innerNode.add( treeNode );
547         getServerInfo().getDiscussions().put( fqDocumentName, new DefaultTreeModel( innerNode ) );
548         // System.out.println( "New topic tree: " + fqDocumentName );
549     }
550 }
551 /**
552 * Setup the group management session and channel and activate it.
553 * Creation date: (03-07-00 11:29:03)
554 */
555 public void createGroupManagement()
556 {
557     Session session = null;
558     Channel groupRequestChannel = null;
559     try
560     {
561         urlString url = urlString.createSessionURL( (String)ServerInfo.getPreferences().getProperty(
562             ServerConstants.getServer_NAME() ),
563             Integer.parseInt( ServerInfo.getPreferences().getProperty(
564                 ServerConstants.getServer_PORT() )),
565             (String)ServerInfo.getPreferences().getProperty( ServerConstants.
566                 getConnection_TYPE() ),
567             (String)ServerConstants.getGROUP_MANAGEMENT_SESSION_NAME() );
568         session = SessionFactory.createSession( getGroupManagementServer(), url, true );
569         groupRequestChannel = session.createChannel( getGroupManagementServer(),
570             ServerConstants.getGROUP_CHANNEL_NAME(),
571             true, true, true );
572         groupRequestChannel.addConsumer( getGroupManagementServer(), getGroupManagementServer() );
573         DataPackage pack = new DataPackage();
574         pack.setContents( null );
575         groupRequestChannel.sendToAll( getGroupManagementServer(), new Data( pack ) );
576         // System.out.println( url.toString() );
577     }
578     catch ( JSDTEException JSDTE )
579     {
580         handleJSDTEException( JSDTE );
581     }
582     getServerInfo().setGroupManagementSession( session );
583     getServerInfo().setGroupManagementChannel( groupRequestChannel );
584 }

```

```

582 /**
583  * Shut down a group.
584  * Creation date: (27-07-00 00:35:12)
585  * @param workGroup peerviewmisc.WorkGroup
586  */
587 public void deactivateGroup(WorkGroup workGroup)
588 {
589     try
590     {
591         urlString url = workGroup.getSessionURL();
592         Session session = SessionFactory.createSession( getGroupManagementServer(), url, false );
593         Channel channel = session.createChannel( getGroupManagementServer(), workGroup.
594             getWorkGroupChannelID(), true, true, false );
595         channel.destroy( getGroupManagementServer() );
596         session.destroy( getGroupManagementServer() );
597     }
598     catch ( JSDTEException JSDTE )
599     {
600         handleJSDTEException( JSDTE );
601     }
602 /**
603  * Insert the method's description here.
604  * Creation date: (20-10-00 15:06:46)
605  */
606 public void deactivateGroups()
607 {
608     Collection values = serverInfo.getGroups().values();
609
610     Iterator iterator = values.iterator();
611     while ( iterator.hasNext() )
612     {
613         deactivateGroup( (WorkGroup) iterator.next() );
614     }
615 }
616 /**
617  * Insert the method's description here.
618  * Creation date: (12-07-00 12:14:10)
619  * @param messageID java.lang.String
620  */
621 public void deleteMessage(String messageID)
622 {
623     serverInfo.getMessages().remove( messageID );
624 }
625 /**
626  * Insert the method's description here.
627  * Creation date: (20-10-00 15:15:34)
628  */
629 public void destroyGroupManagement()
630 {
631     try
632     {
633         getServerInfo().getGroupManagementChannel().destroy( getGroupManagementServer() );
634         getServerInfo().getGroupManagementSession().destroy( getGroupManagementServer() );
635     }
636     catch ( JSDTEException JSDTE )
637     {
638         handleJSDTEException( JSDTE );
639     }
640 }
641 /**
642  * Insert the method's description here.
643  * Creation date: (04-07-00 16:11:19)
644  */
645 protected void finalize()

```

```

648 }
649 /**
650 * Insert the method's description here.
651 * Creation date: (30-06-00 11:17:21)
652 * @return serverapp.PeerViewServer
653 */
654 public PeerViewServer getDefaultServer() {
655     return defaultServer;
656 }
657 /**
658 * Insert the method's description here.
659 * Creation date: (03-07-00 12:23:42)
660 * @return serverapp.GroupManagementServer
661 */
662 public GroupManagementServer getGroupManagementServer()
663 {
664     if (groupManagementServer == null)
665     {
666         groupManagementServer = new GroupManagementServer();
667         groupManagementServer.setName( ServerInfo.getPreferences().getProperty( ServerConstants.
668             SERVER_NAME() ) );
669     }
670     return groupManagementServer;
671 }
672 /**
673 * Insert the method's description here.
674 * Creation date: (02-09-00 08:46:31)
675 * @return serverapp.ServerApplication
676 */
677 public ServerApplication getServerApplication() {
678     return serverApplication;
679 }
680 /**
681 * Insert the method's description here.
682 * Creation date: (28-06-00 22:59:39)
683 * @return serverapp.ServerInfo
684 */
685 public ServerInfo getServerInfo()
686 {
687     if ( serverInfo == null )
688     {
689         serverInfo = new ServerInfo();
690     }
691     return serverInfo;
692 }
693 /**
694 * Handle the different subclasses of JSDTEException, i.e. the different types of exception that JSDT
695 * actions may throw,
696 * by issuing appropriate messages and taking corrective or conclusive action.
697 * Creation date: (08-08-00 16:34:28)
698 * @param exception com.sun.media.jsdt.JSDTEException
699 */
700 public void handleJSDTEException(JSDTEException exception)
701 {
702     String message = (String) ServerConstants.getJSDT_EXCEPTION_MESSAGES_TABLE().get( exception.getClass
703         ().toString() );
704     message += " (" + exception.toString() + ")";
705     System.err.println( message );
706 }
707 /**
708 * Insert the method's description here.
709 * Creation date: (01-07-00 01:14:14)
710 */
711 public void initialize()
712 {

```

```

710 }
711 /**
712 * Read the discussion "database" (a text file, essentially) from disk and initialize the corresponding
713 * data structure.
714 * Creation date: (24-08-00 08:45:43)
715 */
716 public void readDiscussions()
717 {
718     try
719     {
720         FileInputStream fistream = new FileInputStream( ServerConstants.getDISCUSSIONS_FILE_NAME() );
721         ObjectInputStream oistream = new ObjectInputStream( fistream );
722         Hashtable discussionsTable = (Hashtable) oistream.readObject();
723         if ( discussionsTable != null )
724         {
725             getServerInfo().setDiscussions( discussionsTable );
726         }
727         else
728         {
729             getServerInfo().setDiscussions( new Hashtable() );
730         }
731         // System.out.println( "Discussionstablesize: " + discussionsTable.size() );
732         oistream.close();
733     }
734     catch ( Exception e )
735     {
736         System.err.println( ServerConstants.getCOULD_NOT_READ_MESSAGES() );
737     }
738 }
739 /**
740 * Read the groups directory from persistent storage, i.e. disk file
741 * Creation date: (04-07-00 15:03:54)
742 */
743 public void readGroups()
744 {
745     try
746     {
747         FileInputStream fis = null;
748         ObjectInputStream ois = null;
749         fis = new FileInputStream( ServerConstants.getGROUPS_FILE_NAME() );
750         ois = new ObjectInputStream( fis );
751
752         // while ( ois.available() > 0 )
753         // {
754             // the below simulates a loop by trying a readObject operation until an exception is thrown. Not
755             // kosher, but
756             // the only way that seems to work at the moment.
757             while (true)
758             {
759                 try
760                 {
761                     WorkGroup workGroup = (WorkGroup) ois.readObject();
762                     workGroup.setNumberOfDocuments( 0 );
763                     workGroup.setTotalDataVolume( 0 );
764                     workGroup.setDocuments( new java.util.Hashtable() );
765                     workGroup.setParticipants( new java.util.HashSet() );
766                     // System.out.println( "WorkGroup: " + workGroup.getWorkGroupID() );
767                     getServerInfo().getGroups().put( workGroup.getWorkGroupID(), workGroup );
768                     activateGroup( workGroup );
769                 }
770                 catch ( Exception E )
771                 {
772                     break;
773                 }
774             }
775         }
776     }

```

```

774     // }
775     ois.close();
776     broadcastGroupDirectory();
777   }
778   catch ( IOException E )
779   {
780     System.err.println( ServerConstants.getCOULD_NOT_READ_GROUPS() );
781   }
782 }
783 /**
784 * Insert the method's description here.
785 * Creation date: (14-08-00 23:03:40)
786 */
787 public void readMessages()
788 {
789   try
790   {
791     FileInputStream fistream = new FileInputStream( ServerConstants.getMESSAGES_FILE_NAME() );
792     ObjectInputStream oistream = new ObjectInputStream( fistream );
793     Hashtable messageTable = (Hashtable) oistream.readObject();
794     if ( messageTable != null )
795     {
796       getServerInfo().setMessages( messageTable );
797     }
798     else
799     {
800       getServerInfo().setMessages( new Hashtable() );
801     }
802     // System.out.println( "Messagetablesizer: " + messageTable.size() );
803     oistream.close();
804   }
805   catch ( Exception e )
806   {
807     System.err.println( ServerConstants.getCOULD_NOT_READ_MESSAGES() );
808   }
809 }
810 }
811 /**
812 * Insert the method's description here.
813 * Creation date: (27-07-00 00:31:10)
814 * @param workGroup peerviewmisc.WorkGroup
815 */
816 public void removeWorkGroup(WorkGroup workGroup)
817 {
818   System.out.println( "I\u2014removeworkgroup" );
819   deactivateGroup( workGroup );
820   System.out.println( "After\u2014deactivateGroup" );
821   getServerInfo().getGroups().remove( workGroup.getWorkGroupID() );
822   System.out.println( "about\u2014to\u2014broadcast\u2014group\u2014directory\u2014from\u2014removeworkgroup" );
823   broadcastGroupDirectory();
824 }
825 /**
826 * Insert the method's description here.
827 * Creation date: (07-07-00 14:53:16)
828 * @param recipientName java.lang.String
829 */
830 public void sendGroupDirectory(String recipientName)
831 {
832
833   DataPackage outgoingPackage = new DataPackage();
834   outgoingPackage.setContents( serverInfo.getGroups() );
835   outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
836     getGROUP_DIRECTORY() );
837
838   try

```

```

840     {
841         serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer ,
842                                         recipientName ,
843                                         new Data( outgoingPackage ) );
844         System.out.println( "Sent\u00a5group\u00a5directory" );
845     }
846     catch ( JSDTE exception JSDTE )
847     {
848         handleJSDTEException( JSDTE );
849     }
850 }
852 /**
853 * Insert the method's description here.
854 * Creation date: (10-07-00 02:20:49)
855 * @param messageID java.lang.String
856 * @param recipientName java.lang.String
857 */
858 public void sendMessage(String messageID, String recipientName)
859 {
860     DataPackage outgoingPackage = new DataPackage();
861
862     // System.out.println( "Sent message: " + messageID );
863     outgoingPackage.setContents( serverInfo.getMessages().get( messageID ) );
864     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.getMessage()
865                                         () );
866     try
867     {
868         serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer, recipientName, new
869                                         Data( outgoingPackage ) );
870     }
871     catch ( JSDTE exception JSDTE )
872     {
873         handleJSDTEException( JSDTE );
874     }
875 /**
876 * Precondition: documentName is the fully qualified name of a document as specified in method
877 * SharedConstants.createFullyQualifiedDocumentName
878 * Creation date: (09-07-00 22:20:13)
879 * @param documentName java.lang.String
880 * @param recipientName java.lang.String
881 */
882 public void sendTopicTree(String documentName, String recipientName)
883 {
884     DataPackage outgoingPackage = new DataPackage();
885
886     // System.out.println( "Sending topic tree: " + documentName );
887     outgoingPackage.setContents( getServerInfo().getDiscussions().get( documentName ) );
888     outgoingPackage.getProperties().setProperty( DataPackage.getDESCRIPTION(), ServerConstants.
889                                         getTOPIC_TREE() );
890     outgoingPackage.getProperties().setProperty( DataPackage.getDOC_PATH(), documentName );
891     try
892     {
893         serverInfo.getGroupManagementChannel().sendToClient( groupManagementServer, recipientName, new
894                                         Data( outgoingPackage ) );
895     }
896     catch ( JSDTE exception JSDTE )
897     {
898         handleJSDTEException( JSDTE );
899     }
900 }
901 /**
902 * Insert the method's description here.
903 * Creation date: (30-06-00 11:17:21)

```

```

900 * @param newDefaultServer serverapp.PeerViewServer
901 */
902 public void setDefaultServer(PeerViewServer newDefaultServer) {
903     defaultServer = newDefaultServer;
904 }
905 /**
906 * Insert the method's description here.
907 * Creation date: (03-07-00 12:23:42)
908 * @param newGroupManagementServer serverapp.GroupManagementServer
909 */
910 public void setGroupManagementServer(GroupManagementServer newGroupManagementServer) {
911     groupManagementServer = newGroupManagementServer;
912 }
913 /**
914 * Insert the method's description here.
915 * Creation date: (02-09-00 08:46:31)
916 * @param newServerApplication serverapp.ServerApplication
917 */
918 public void setServerApplication(ServerApplication newServerApplication) {
919     serverApplication = newServerApplication;
920 }
921 /**
922 * Insert the method's description here.
923 * Creation date: (28-06-00 22:59:39)
924 * @param newServerInfo serverapp.ServerInfo
925 */
926 public void setServerInfo(ServerInfo newServerInfo) {
927     serverInfo = newServerInfo;
928 }
929 /**
930 * Insert the method's description here.
931 * Creation date: (27-06-00 20:18:18)
932 */
933 public void startRegistry()
934 {
935     try
936     {
937         RegistryFactory.registryPort = Integer.parseInt( ServerInfo.getPreferences().getProperty(
938             ServerConstants.getREGISTRY_PORT() ));
939         com.sun.media.jsdt.RegistryFactory.startRegistry(
940             (String) ServerInfo.getPreferences().getProperty( ServerConstants.getConnection_TYPE() ));
941         // Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.getREGISTRY_PORT()
942         //));
943         if ( RegistryFactory.registryExists( ServerInfo.getPreferences().getProperty( ServerConstants.
944             getConnection_TYPE() ),
945             Integer.parseInt( ServerInfo.getPreferences().getProperty(
946                 ServerConstants.getREGISTRY_PORT() ))));
947         {
948             System.out.println( "Successfully started registry" );
949         }
950     }
951     catch ( com.sun.media.jsdt.NoRegistryException nre )
952     {
953         // System.out.println( (String) ServerInfo.getPreferences().getProperty( ServerConstants.
954             getConnection_TYPE() ));
955         // System.out.println( ServerInfo.getPreferences().getProperty( ServerConstants.getREGISTRY_PORT()
956             //));
957         System.err.println( nre.toString() );
958         System.exit(0);
959     }
960     catch ( com.sun.media.jsdt.RegistryExistsException ree )
961     {
962         System.err.println( ree.toString() );
963         System.exit(0);
964     }

```

```

960 }
962 /**
963 * Insert the method's description here.
964 * Creation date: (21-10-00 23:27:34)
965 */
966 public void stopRegistry()
967 {
968     try
969     {
970         RegistryFactory.stopRegistry( ServerInfo.getPreferences().getProperty( ServerConstants.
971             getConnection_Type() ),
972             Integer.parseInt( ServerInfo.getPreferences().getProperty( ServerConstants.
973                 getServer_Port() ) ) );
974     }
975     catch ( JSDTEException JSDTE )
976     {
977         handleJSDTEException( JSDTE );
978     }
979 /**
980 * Insert the method's description here.
981 * Creation date: (21-10-00 23:20:36)
982 */
983 public void terminate()
984 {
985     deactivateGroups();
986     writeDiscussions();
987     writeMessages();
988     writeGroups();
989     getServerInfo().writePreferences();
990     destroyGroupManagement();
991     try
992     {
993         System.runFinalization();
994         Runtime.getRuntime().exit( 0 );
995     }
996     catch ( Exception E )
997     {
998         System.out.println( ServerConstants.getCOULD_NOT_EXIT_PROPERLY() );
999     }
1000 /**
1001 * Insert the method's description here.
1002 * Creation date: (11-07-00 21:37:44)
1003 * @param documentName java.lang.String
1004 * @param newTree javax.swing.tree.DefaultTreeModel
1005 */
1006 public void updateTopicTree(String documentName, DefaultTreeModel newTree)
1007 {
1008     getServerInfo().getDiscussions().put( documentName, newTree );
1009     broadcastTopicTree( documentName, newTree );
1010 }
1011 /**
1012 * Insert the method's description here.
1013 * Creation date: (22-07-00 18:40:42)
1014 * @param workGroup peerviewmisc.WorkGroup
1015 */
1016 public void updateWorkGroup(WorkGroup workGroup)
1017 {
1018     WorkGroup groupToUpdate = (WorkGroup) getServerInfo().getGroups().get( workGroup.getWorkGroupID()
1019     );
1020     groupToUpdate.setNumberOfDocuments( groupToUpdate.getNumberOfDocuments() + workGroup.
1021         getNumberOfDocuments() );

```

```

        groupToBeUpdated.setTotalDataVolume( groupToBeUpdated.getTotalDataVolume() + workGroup.
            getTotalDataVolume() );
1022    getServerInfo().getGroups().put( groupToBeUpdated.getWorkGroupID(), groupToBeUpdated );
        broadcastGroupDirectory();
1024 }
1025 /**
1026 * Insert the method's description here.
1027 * Creation date: (24-08-00 08:39:39)
1028 */
1029 public void writeDiscussions()
1030 {
1031     try
1032     {
1033         FileOutputStream fostream = new FileOutputStream( ServerConstants.getDISCUSSIONS_FILE_NAME() );
1034         ObjectOutputStream oostream = new ObjectOutputStream( fostream );
1035         oostream.writeObject( getServerInfo().getDiscussions() );
1036         // System.out.println( "WroteDiscussions" );
1037         oostream.flush();
1038         oostream.close();
1039     }
1040     catch ( Exception e )
1041     {
1042         System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
1043     }
1044 }
1045 /**
1046 * Write registered groups to persistent storage, i.e. disk file
1047 * Creation date: (04-07-00 15:03:41)
1048 */
1049 public void writeGroups()
1050 {
1051     try
1052     {
1053         FileOutputStream fos = new java.io.FileOutputStream( ServerConstants.getGROUPS_FILE_NAME() );
1054         ObjectOutputStream oos = new ObjectOutputStream( fos );
1055
1056         Collection values = serverInfo.getGroups().values();
1057         Iterator iterator = values.iterator();
1058
1059         while ( iterator.hasNext() )
1060         {
1061             WorkGroup workGroup = (WorkGroup) iterator.next();
1062             workGroup.setParticipants( null );
1063             workGroup.setDocuments( null );
1064             oos.writeObject( workGroup );
1065             /* oos.writeUTF( workGroup.getGroupName() );
1066             oos.writeInt( workGroup.getMaximumParticipants() );
1067             oos.writeUTF( workGroup.getDescription() );
1068             oos.writeUTF( workGroup.getCreationDate().toString() );
1069             oos.writeUTF( workGroup.getCreator() );
1070             */
1071             // System.out.println( "Wrote group" );
1072         }
1073         oos.flush();
1074         oos.close();
1075     }
1076     catch ( IOException IOE )
1077     {
1078         System.err.println( ServerConstants.getCOULD_NOT_WRITE_GROUPS() );
1079     }
1080 }
1081 /**
1082 * Insert the method's description here.
1083 * Creation date: (14-08-00 23:26:34)

```

```

1086     */
1087     public void writeMessages()
1088     {
1089         try
1090         {
1091             FileOutputStream fostream = new FileOutputStream( ServerConstants.getMESSAGES_FILE_NAME() );
1092             ObjectOutputStream oostream = new ObjectOutputStream( fostream );
1093             oostream.writeObject( getServerInfo().getMessages() );
1094             // System.out.println( "WroteMessages" );
1095             oostream.flush();
1096             oostream.close();
1097         }
1098         catch ( Exception e )
1099         {
1100             System.err.println( ServerConstants.getCOULD_NOT_WRITE_MESSAGES() );
1101         }
1102     }
1103 }
```

Listing C.33: ServerSetup.java

```

package serverapp;
2
4 import javax.swing.*;
4 import javax.swing.border.*;
6 import javax.swing.event.*;
6 import java.awt.*;
8 import java.util.Properties;
8 import java.io.*;
import java.awt.event.ActionListener;
10 import java.awt.event.*;
/*
12 * The server setup dialog box.
12 * Creation date: (28-05-00 20:58:57)
14 * @author:
14 */
16 public class ServerSetup extends JDialog
{
18     public class OKButtonHandler implements ActionListener
18     {
20         public void actionPerformed( ActionEvent ae )
20         {
22             submitAndClose();
22         }
24     };
26
26     public class CancelButtonHandler implements ActionListener
26     {
28         public void actionPerformed( ActionEvent ae )
28         {
30             cancelAndClose();
30         }
32     };
34
34     private JButton ivjCancelButton = null;
36     private JPanel ivjJDialogContentPane = null;
36     private JPanel ivj JPanel1 = null;
38     private java.awt.FlowLayout ivj JPanel1FlowLayout = null;
38     private JButton ivjOKButton = null;
40     private JComboBox ivjProtocolComboBox = null;
40     private JLabel ivjProtocolLabel = null;
42     private JTextField ivjServerNameField = null;
42     private JLabel ivjServerNameLabel = null;
44     private JLabel ivjServerPortLabel = null;
44     private JTextField ivjServerPortField = null;
46     private JPanel ivjTopPanel = null;
```

```

    /**
48     * ServerSetup constructor comment.
49     */
50 public ServerSetup() {
51     super();
52     initialize();
53 }
54 /**
55     * ServerSetup constructor comment.
56     * @param owner java.awt.Dialog
57     */
58 public ServerSetup(java.awt.Dialog owner) {
59     super(owner);
60 }
61 /**
62     * ServerSetup constructor comment.
63     * @param owner java.awt.Dialog
64     * @param title java.lang.String
65     */
66 public ServerSetup(java.awt.Dialog owner, String title) {
67     super(owner, title);
68 }
69 /**
70     * ServerSetup constructor comment.
71     * @param owner java.awt.Dialog
72     * @param title java.lang.String
73     * @param modal boolean
74     */
75 public ServerSetup(java.awt.Dialog owner, String title, boolean modal) {
76     super(owner, title, modal);
77 }
78 /**
79     * ServerSetup constructor comment.
80     * @param owner java.awt.Dialog
81     * @param modal boolean
82     */
83 public ServerSetup(java.awt.Dialog owner, boolean modal) {
84     super(owner, modal);
85 }
86 /**
87     * ServerSetup constructor comment.
88     * @param owner java.awt.Frame
89     */
90 public ServerSetup(java.awt.Frame owner) {
91     super(owner);
92 }
93 /**
94     * ServerSetup constructor comment.
95     * @param owner java.awt.Frame
96     * @param title java.lang.String
97     */
98 public ServerSetup(java.awt.Frame owner, String title) {
99     super(owner, title);
100 }
101 /**
102     * ServerSetup constructor comment.
103     * @param owner java.awt.Frame
104     * @param title java.lang.String
105     * @param modal boolean
106     */
107 public ServerSetup(java.awt.Frame owner, String title, boolean modal) {
108     super(owner, title, modal);
109 }
110 /**
111     * ServerSetup constructor comment.
112     * @param owner java.awt.Frame

```

```

114     * @param modal boolean
115     */
116     public ServerSetup(java.awt.Frame owner, boolean modal) {
117         super(owner, modal);
118     }
119 /**
120  * Insert the method's description here.
121  * Creation date: (05-08-00 20:54:45)
122  */
123 public void cancelAndClose()
124 {
125     dispose();
126 }
127 /**
128  * Return the CancelButton property value.
129  * @return javax.swing.JButton
130  */
131 /* WARNING: THIS METHOD WILL BE REGENERATED. */
132 private javax.swing.JButton getCancelButton() {
133     if (ivjCancelButton == null) {
134         try {
135             ivjCancelButton = new javax.swing.JButton();
136             ivjCancelButton.setName("CancelButton");
137             ivjCancelButton.setText("Cancel");
138             // user code begin {1}
139             ivjCancelButton.addActionListener( new CancelButtonHandler() );
140             // user code end
141             handleException(ivjExc);
142         } catch (java.lang.Throwable ivjExc) {
143             // user code begin {2}
144             // user code end
145             handleException(ivjExc);
146         }
147     }
148     return ivjCancelButton;
149 }
150 /**
151  * Return the JDialogContentPane property value.
152  * @return javax.swing.JPanel
153  */
154 /* WARNING: THIS METHOD WILL BE REGENERATED. */
155 private javax.swing.JPanel getJDialogContentPane() {
156     if (ivjJDialogContentPane == null) {
157         try {
158             ivjJDialogContentPane = new javax.swing.JPanel();
159             ivjJDialogContentPane.setName("JDialogContentPane");
160             ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
161             getJDialogContentPane().add(get JPanel1(), "South");
162             getJDialogContentPane().add(get JPanel1(), "Center");
163             // user code begin {1}
164             // user code end
165             handleException(ivjExc);
166         }
167     }
168     return ivjJDialogContentPane;
169 }
170 /**
171  * Return the JPanel1 property value.
172  * @return javax.swing.JPanel
173  */
174 /* WARNING: THIS METHOD WILL BE REGENERATED. */
175 private javax.swing.JPanel get JPanel1() {
176     if (ivj JPanel1 == null) {
177         try {

```

```

        ivj JPanel1 = new javax.swing.JPanel();
180    ivj JPanel1.setName("JPanel1");
181    ivj JPanel1.setLayout(get JPanel1FlowLayout());
182    get JPanel1().add(getOKButton(), getOKButton().getName());
183    get JPanel1().add(getCancelButton(), getCancelButton().getName());
184    // user code begin {1}
185    // user code end
186    } catch (java.lang.Throwable ivjExc) {
187        // user code begin {2}
188        // user code end
189        handleException(ivjExc);
190    }
191 }
192 /**
193 * Return the JPanel1FlowLayout property value.
194 * @return java.awt.FlowLayout
195 */
196 /* WARNING: THIS METHOD WILL BE REGENERATED. */
197 private java.awt.FlowLayout get JPanel1FlowLayout() {
200    java.awt.FlowLayout ivj JPanel1FlowLayout = null;
201    try {
202        /* Create part */
203        ivj JPanel1FlowLayout = new java.awt.FlowLayout();
204        ivj JPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
205    } catch (java.lang.Throwable ivjExc) {
206        handleException(ivjExc);
207    };
208    return ivj JPanel1FlowLayout;
209 }
210 /**
211 * Return the OKButton property value.
212 * @return javax.swing.JButton
213 */
214 /* WARNING: THIS METHOD WILL BE REGENERATED. */
215 private javax.swing.JButton getOKButton() {
216    if (ivjOKButton == null) {
217        try {
218            ivjOKButton = new javax.swing.JButton();
219            ivjOKButton.setName("OKButton");
220            ivjOKButton.setText("OK");
221            // user code begin {1}
222            ivjOKButton.addActionListener( new OKButtonHandler() );
223            // user code end
224        } catch (java.lang.Throwable ivjExc) {
225            // user code begin {2}
226            // user code end
227            handleException(ivjExc);
228        }
229    }
230    return ivjOKButton;
231 }
232 /**
233 * Return the ProtocolComboBox property value.
234 * @return javax.swing.JComboBox
235 */
236 /* WARNING: THIS METHOD WILL BE REGENERATED. */
237 private javax.swing.JComboBox getProtocolComboBox() {
238    if (ivjProtocolComboBox == null) {
239        try {
240            ivjProtocolComboBox = new javax.swing.JComboBox();
241            ivjProtocolComboBox.setName("ProtocolComboBox");
242            // user code begin {1}
243            // user code end
244        } catch (java.lang.Throwable ivjExc) {

```

```

246         // user code begin {2}
247         // user code end
248         handleException(ivjExc);
249     }
250     return ivjProtocolComboBox;
251 }
252 /**
253 * Return the ProtocolLabel property value.
254 * @return javax.swing.JLabel
255 */
256 /* WARNING: THIS METHOD WILL BE REGENERATED. */
257 private javax.swing.JLabel getProtocolLabel() {
258     if (ivjProtocolLabel == null) {
259         try {
260             ivjProtocolLabel = new javax.swing.JLabel();
261             ivjProtocolLabel.setName("ProtocolLabel");
262             ivjProtocolLabel.setText("Connection:");
263             // user code begin {1}
264             // user code end
265         } catch (java.lang.Throwable ivjExc) {
266             // user code begin {2}
267             // user code end
268             handleException(ivjExc);
269         }
270     }
271     return ivjProtocolLabel;
272 }
273 /**
274 * Return the ServerNameField property value.
275 * @return javax.swing.JTextField
276 */
277 /* WARNING: THIS METHOD WILL BE REGENERATED. */
278 private javax.swing.JTextField getServerNameField() {
279     if (ivjServerNameField == null) {
280         try {
281             ivjServerNameField = new javax.swing.JTextField();
282             ivjServerNameField.setName("ServerNameField");
283             // user code begin {1}
284             // user code end
285         } catch (java.lang.Throwable ivjExc) {
286             // user code begin {2}
287             // user code end
288             handleException(ivjExc);
289         }
290     }
291     return ivjServerNameField;
292 }
293 /**
294 * Return the ServerNameLabel property value.
295 * @return javax.swing.JLabel
296 */
297 /* WARNING: THIS METHOD WILL BE REGENERATED. */
298 private javax.swing.JLabel getServerNameLabel() {
299     if (ivjServerNameLabel == null) {
300         try {
301             ivjServerNameLabel = new javax.swing.JLabel();
302             ivjServerNameLabel.setName("ServerNameLabel");
303             ivjServerNameLabel.setText("Server\u00d7name:");
304             // user code begin {1}
305             // user code end
306         } catch (java.lang.Throwable ivjExc) {
307             // user code begin {2}
308             // user code end
309             handleException(ivjExc);
310         }
311     }

```

```

        }
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) {
321         try {
322             ivjServerPortField = new javax.swing.JTextField();
323             ivjServerPortField.setName("ServerPortField");
324             // user code begin {1}
325             // user code end
326         } catch (java.lang.Throwable ivjExc) {
327             // user code begin {2}
328             // user code end
329             handleException(ivjExc);
330         }
331     }
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) {
341         try {
342             ivjServerPortLabel = new javax.swing.JLabel();
343             ivjServerPortLabel.setName("ServerPortLabel");
344             ivjServerPortLabel.setText("ServerPort:");
345             // user code begin {1}
346             // user code end
347         } catch (java.lang.Throwable ivjExc) {
348             // user code begin {2}
349             // user code end
350             handleException(ivjExc);
351         }
352     }
353     return ivjServerPortLabel;
}
354 /**
 * Return the JPanel2 property value.
356 * @return javax.swing.JPanel
 */
358 /* WARNING: THIS METHOD WILL BE REGENERATED. */
360 private javax.swing.JPanel getTopPanel() {
361     if (ivjTopPanel == null) {
362         try {
363             ivjTopPanel = new javax.swing.JPanel();
364             ivjTopPanel.setName("TopPanel");
365             ivjTopPanel.setLayout(new java.awt.GridBagLayout());
366
367             java.awt.GridBagConstraints constraintsServerNameLabel = new java.awt.GridBagConstraints();
368             constraintsServerNameLabel.gridx = 0; constraintsServerNameLabel.gridy = 0;
369             constraintsServerNameLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
370             constraintsServerNameLabel.weighty = 0.2;
371             constraintsServerNameLabel.insets = new java.awt.Insets(6, 6, 6, 6);
372             getTopPanel().add(getServerNameLabel(), constraintsServerNameLabel);
373
374             java.awt.GridBagConstraints constraintsServerNameField = new java.awt.GridBagConstraints();
375             constraintsServerNameField.gridx = 1; constraintsServerNameField.gridy = 0;
376             constraintsServerNameField.fill = java.awt.GridBagConstraints.HORIZONTAL;

```

```

378     constraintsServerNameField.anchor = java.awt.GridBagConstraints.NORTHEAST;
379     constraintsServerNameField.weightx = 1.0;
380     constraintsServerNameField.weighty = 0.2;
381     constraintsServerNameField.insets = new java.awt.Insets(6, 6, 6, 6);
382     getTopPanel().add(getServerNameField(), constraintsServerNameField);
383
384     java.awt.GridBagConstraints constraintsServerPortLabel = new java.awt.GridBagConstraints();
385     constraintsServerPortLabel.gridx = 0; constraintsServerPortLabel.gridy = 1;
386     constraintsServerPortLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
387     constraintsServerPortLabel.weighty = 0.2;
388     constraintsServerPortLabel.insets = new java.awt.Insets(6, 6, 6, 6);
389     getTopPanel().add(getServerPortLabel(), constraintsServerPortLabel);
390
390     java.awt.GridBagConstraints constraintsServerPortField = new java.awt.GridBagConstraints();
391     constraintsServerPortField.gridx = 1; constraintsServerPortField.gridy = 1;
392     constraintsServerPortField.anchor = java.awt.GridBagConstraints.NORTHWEST;
393     constraintsServerPortField.weightx = 1.0;
394     constraintsServerPortField.weighty = 0.2;
395     constraintsServerPortField.ipadx = 100;
396     constraintsServerPortField.insets = new java.awt.Insets(6, 6, 6, 6);
397     getTopPanel().add(getServerPortField(), constraintsServerPortField);
398
398     java.awt.GridBagConstraints constraintsProtocolLabel = new java.awt.GridBagConstraints();
399     constraintsProtocolLabel.gridx = 0; constraintsProtocolLabel.gridy = 2;
400     constraintsProtocolLabel.anchor = java.awt.GridBagConstraints.NORTHWEST;
401     constraintsProtocolLabel.weighty = 0.2;
402     constraintsProtocolLabel.insets = new java.awt.Insets(6, 6, 6, 6);
403     getTopPanel().add(getProtocolLabel(), constraintsProtocolLabel);
404
406     java.awt.GridBagConstraints constraintsProtocolComboBox = new java.awt.GridBagConstraints();
407     constraintsProtocolComboBox.gridx = 1; constraintsProtocolComboBox.gridy = 2;
408     constraintsProtocolComboBox.anchor = java.awt.GridBagConstraints.NORTHWEST;
409     constraintsProtocolComboBox.weightx = 1.0;
410     constraintsProtocolComboBox.weighty = 0.2;
411     constraintsProtocolComboBox.insets = new java.awt.Insets(6, 6, 6, 6);
412     getTopPanel().add(getProtocolComboBox(), constraintsProtocolComboBox);
413     // user code begin {1}
414     Border matteBorder = BorderFactory.createMatteBorder(6,6,6,6,ivjTopPanel.getBackground());
415     Border lineBorder = BorderFactory.createLineBorder(Color.gray);
416     Border compoundBorder = BorderFactory.createCompoundBorder(lineBorder, matteBorder);
417     Border secondCompoundBorder = BorderFactory.createCompoundBorder(matteBorder, compoundBorder);
418     ivjTopPanel.setBorder(secondCompoundBorder);
419     // user code end
420   } catch (java.lang.Throwable ivjExc) {
421     // user code begin {2}
422     // user code end
423     handleException(ivjExc);
424   }
425 }
426 return ivjTopPanel;
427 }
428 /**
429 * Called whenever the part throws an exception.
430 * @param exception java.lang.Throwable
431 */
432 private void handleException(java.lang.Throwable exception) {
433
434   /* Uncomment the following lines to print uncaught exceptions to stdout */
435   // System.out.println("----- UNCAUGHT EXCEPTION -----");
436   // exception.printStackTrace(System.out);
437 }
438 /**
439 * Initialize the class.
440 */
441 /* WARNING: THIS METHOD WILL BE REGENERATED. */
442 private void initialize() {

```

```

try {
    // 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 {
        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);
            }
        });
        aServerSetup.setVisible(true);
    } catch (Throwable exception) {
474        System.err.println("Exception occurred in main() of javax.swing.JDialog");
        exception.printStackTrace(System.out);
    }
}
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 */
500 public void submitAndClose()
{
    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

```

  * @param owner java.awt.Frame
60   */
61   public AboutDialog(java.awt.Frame owner) {
62     super(owner);
63   }
64 /**
65  * AboutDialog constructor comment.
66  * @param owner java.awt.Frame
67  * @param title java.lang.String
68  */
69   public AboutDialog(java.awt.Frame owner, String title) {
70     super(owner, title);
71   }
72 /**
73  * AboutDialog constructor comment.
74  * @param owner java.awt.Frame
75  * @param title java.lang.String
76  * @param modal boolean
77  */
78   public AboutDialog(java.awt.Frame owner, String title, boolean modal) {
79     super(owner, title, modal);
80   }
81 /**
82  * AboutDialog constructor comment.
83  * @param owner java.awt.Frame
84  * @param modal boolean
85  */
86   public AboutDialog(java.awt.Frame owner, boolean modal) {
87     super(owner, modal);
88   }
89 /**
90  * Return the JDialoContentPane property value.
91  * @return javax.swing.JPanel
92  */
93 /* WARNING: THIS METHOD WILL BE REGENERATED. */
94 private javax.swing.JPanel getJDialoContentPane() {
95   if (ivjJDialoContentPane == null) {
96     try {
97       ivjJDialoContentPane = new javax.swing.JPanel();
98       ivjJDialoContentPane.setName("JDialoContentPane");
99       ivjJDialoContentPane.setLayout(new java.awt.GridBagLayout());
100      ivjJDialoContentPane.setBackground(java.awt.Color.white);

101      java.awt.GridBagConstraints constraintsLogoLabel = new java.awt.GridBagConstraints();
102      constraintsLogoLabel.gridx = 0; constraintsLogoLabel.gridy = 0;
103      constraintsLogoLabel.gridwidth = 2;
104      constraintsLogoLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
105      constraintsLogoLabel.weightx = 1.0;
106      constraintsLogoLabel.ipady = 5;
107      constraintsLogoLabel.insets = new java.awt.Insets(5, 5, 0, 5);
108      getJDialoContentPane().add(getLogoLabel(), constraintsLogoLabel);
109

110      java.awt.GridBagConstraints constraintsRulerLabel = new java.awt.GridBagConstraints();
111      constraintsRulerLabel.gridx = 0; constraintsRulerLabel.gridy = 1;
112      constraintsRulerLabel.gridwidth = 3;
113      constraintsRulerLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
114      constraintsRulerLabel.ipady = 10;
115      constraintsRulerLabel.insets = new java.awt.Insets(0, 5, 5, 5);
116      getJDialoContentPane().add(getRulerLabel(), constraintsRulerLabel);
117

118      java.awt.GridBagConstraints constraintsTextLabel = new java.awt.GridBagConstraints();
119      constraintsTextLabel.gridx = 0; constraintsTextLabel.gridy = 2;
120      constraintsTextLabel.gridwidth = 2;
121      constraintsTextLabel.fill = java.awt.GridBagConstraints.HORIZONTAL;
122      constraintsTextLabel.weightx = 1.0;
123      constraintsTextLabel.ipady = 10;

```

```

126     constraintsTextLabel.insets = new java.awt.Insets(5, 5, 5, 5);
127     getJDialogContentPane().add(getTextLabel(), constraintsTextLabel);
128     // user code begin {1}
129     // user code end
130 } catch (java.lang.Throwable ivjExc) {
131     // user code begin {2}
132     // user code end
133     handleException(ivjExc);
134 }
135 return ivjJDialogContentPane;
136 }
137 /**
138 * Return the LogoLabel property value.
139 * @return javax.swing.JLabel
140 */
141 /* WARNING: THIS METHOD WILL BE REGENERATED. */
142 private javax.swing.JLabel getLogoLabel() {
143     if (ivjLogoLabel == null) {
144         try {
145             ivjLogoLabel = new javax.swing.JLabel();
146             ivjLogoLabel.setName("LogoLabel");
147             ivjLogoLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/peerviewlogo2.gif")));
148             ivjLogoLabel.setText("");
149             ivjLogoLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
150             ivjLogoLabel.setVerticalTextPosition(javax.swing.SwingConstants.CENTER);
151             // user code begin {1}
152             /* Border raisedBorder = BorderFactory.createEtchedBorder( new java.awt.Color( 33, 132, 132 ), new
153             java.awt.Color( 189, 189, 189 ) );
154
155             Border matteBorder = BorderFactory.createMatteBorder(6,6,6,6,ivjLogoLabel.getBackground() );
156             Border compoundBorder = BorderFactory.createCompoundBorder(matteBorder, raisedBorder);
157             ivjLogoLabel.setBorder(compoundBorder);
158             */
159             ivjLogoLabel.setBackground( java.awt.Color.white );
160             ivjLogoLabel.setForeground( java.awt.Color.white );
161             ivjLogoLabel.setVerticalTextPosition( SwingConstants.BOTTOM );
162             // ivjLogoLabel.setVerticalAlignment( SwingConstants.BOTTOM );
163             ivjLogoLabel.setText( "Release\u20220.900\u2022(beta)" );
164             // user code end
165         } catch (java.lang.Throwable ivjExc) {
166             // user code begin {2}
167             // user code end
168             handleException(ivjExc);
169         }
170     }
171     return ivjLogoLabel;
172 }
173 /**
174 * Return the RulerLabel property value.
175 * @return javax.swing.JLabel
176 */
177 /* WARNING: THIS METHOD WILL BE REGENERATED. */
178 private javax.swing.JLabel getRulerLabel() {
179     if (ivjRulerLabel == null) {
180         try {
181             ivjRulerLabel = new javax.swing.JLabel();
182             ivjRulerLabel.setName("RulerLabel");
183             ivjRulerLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ruler2.gif")));
184             ivjRulerLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
185             ivjRulerLabel.setText("");
186             ivjRulerLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
187             // user code begin {1}
188             // user code end
189         } catch (java.lang.Throwable ivjExc) {
190             // user code begin {2}

```

```

190         // user code end
191         handleException(ivjExc);
192     }
193 }
194 return ivjRulerLabel;
}
195 /**
196 * Return the TextLabel property value.
197 * @return javax.swing.JLabel
198 */
199 /* WARNING: THIS METHOD WILL BE REGENERATED. */
200 private javax.swing.JLabel getTextLabel() {
201     if (ivjTextLabel == null) {
202         try {
203             ivjTextLabel = new javax.swing.JLabel();
204             ivjTextLabel.setName("TextLabel");
205             ivjTextLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/text.gif")));
206             ivjTextLabel.setAlignmentX(java.awt.Component.CENTER_ALIGNMENT);
207             ivjTextLabel.setText("");
208             ivjTextLabel.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
209             // user code begin {1}
210             // user code end
211         } catch (java.lang.Throwable ivjExc) {
212             // user code begin {2}
213             // user code end
214             handleException(ivjExc);
215         }
216     }
217     return ivjTextLabel;
}
218 }
219 /**
220 * Called whenever the part throws an exception.
221 * @param exception java.lang.Throwable
222 */
223 private void handleException(java.lang.Throwable exception) {
224
225     /* Uncomment the following lines to print uncaught exceptions to stdout */
226     // System.out.println("----- UNCAUGHT EXCEPTION -----");
227     // exception.printStackTrace(System.out);
}
228 /**
229 * Initialize the class.
230 */
231 /* WARNING: THIS METHOD WILL BE REGENERATED. */
232 private void initialize() {
233     try {
234         // user code begin {1}
235         // user code end
236         setName("AboutDialog");
237         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
238         setTitle("AboutPeerView");
239         setBackground(java.awt.SystemColor.window);
240         setModal(true);
241         setSize(483, 267);
242         setResizable(false);
243         setContentPane(getJDialogContentPane());
244     } catch (java.lang.Throwable ivjExc) {
245         handleException(ivjExc);
246     }
247     // user code begin {2}
248     setLocationRelativeTo( getOwner() );
249     // user code end
250 }
251 /**
252 * main entrypoint - starts the part when it is run as an application
253 * @param args java.lang.String[]

```

```

256  */
257  public static void main(java.lang.String[] args) {
258      try {
259          AboutDialog aAboutDialog;
260          aAboutDialog = new AboutDialog();
261          aAboutDialog.setModal(true);
262          aAboutDialog.addWindowListener(new java.awt.event.WindowAdapter() {
263              public void windowClosing(java.awt.event.WindowEvent e) {
264                  System.exit(0);
265              }
266          });
267          aAboutDialog.setVisible(true);
268      } catch (Throwable exception) {
269          System.err.println("Exception occurred in main() of javax.swing.JDialog");
270          exception.printStackTrace(System.out);
271      }
272  }

```

Listing C.35: CompressedDocumentPackage.java

```

package peerviewmisc;
1
import java.io.Serializable;
2
3 /**
4  * This class implements a container for transmitting compressed byte arrays and their descriptors
5  * around a network using
6  * the Java serialization mechanism for marshalling and unmarshalling.
7  * Creation date: (19-08-00 13:07:49)
8  * @author:
9  */
10
public class CompressedDocumentPackage implements Serializable
11{
12    private int uncompressedLength;
13    private int compressedLength;
14    private byte[] compressedContents = null;
15    private java.lang.String path = null;
16    private java.lang.String name = null;
17
18 /**
19  * CompressedDocumentPackage constructor comment.
20  */
21    public CompressedDocumentPackage() {
22        super();
23    }
24 /**
25  * Insert the method's description here.
26  * Creation date: (19-08-00 13:10:51)
27  * @return byte[]
28  */
29    public byte[] getCompressedContents() {
30        return compressedContents;
31    }
32 /**
33  * Insert the method's description here.
34  * Creation date: (19-08-00 13:08:45)
35  * @return int
36  */
37    public int getCompressedLength() {
38        return compressedLength;
39    }
40 /**
41  * Insert the method's description here.
42  * Creation date: (28-10-00 00:18:50)
43  * @return java.lang.String
44  */
45    public java.lang.String getName() {

```

```

46     return name;
47 }
48 /**
49 * Insert the method's description here.
50 * Creation date: (05-09-00 14:24:28)
51 * @return java.lang.String
52 */
53 public java.lang.String getPath() {
54     return path;
55 }
56 /**
57 * Insert the method's description here.
58 * Creation date: (19-08-00 13:08:13)
59 * @return int
60 */
61 public int getUncompressedLength() {
62     return uncompressedLength;
63 }
64 /**
65 * Insert the method's description here.
66 * Creation date: (19-08-00 13:10:51)
67 * @param newCompressedContents byte[]
68 */
69 public void setCompressedContents(byte[] newCompressedContents) {
70     compressedContents = newCompressedContents;
71 }
72 /**
73 * Insert the method's description here.
74 * Creation date: (19-08-00 13:08:45)
75 * @param newCompressedLength int
76 */
77 public void setCompressedLength(int newCompressedLength) {
78     compressedLength = newCompressedLength;
79 }
80 /**
81 * Insert the method's description here.
82 * Creation date: (28-10-00 00:18:50)
83 * @param newName java.lang.String
84 */
85 public void setName(java.lang.String newName) {
86     name = newName;
87 }
88 /**
89 * Insert the method's description here.
90 * Creation date: (05-09-00 14:24:28)
91 * @param newPath java.lang.String
92 */
93 public void setPath(java.lang.String newPath) {
94     path = newPath;
95 }
96 /**
97 * Insert the method's description here.
98 * Creation date: (19-08-00 13:08:13)
99 * @param newUncompressedLength int
100 */
101 public void setUncompressedLength(int newUncompressedLength) {
102     uncompressedLength = newUncompressedLength;
103 }
104 }

```

Listing C.36: ConfirmationBox.java

```

package peerviewmisc;
2
3 import javax.swing.event.*;
4 import java.awt.event.*;

```

```

6  /**
7   * This class implements the confirmation box displayed when a user is asked to confirm or deny an
8   * action.
9   * Creation date: (12-07-00 11:11:22)
10  * @author:
11  */
12
13  public class ConfirmationBox extends javax.swing.JDialog {
14
15      public class OKButtonHandler implements ActionListener
16      {
17          public void actionPerformed( ActionEvent ae )
18          {
19              submitAndClose();
20          }
21      }
22
23      public class CancelButtonHandler implements ActionListener
24      {
25          public void actionPerformed( ActionEvent ae )
26          {
27              cancelAndClose();
28          }
29      }
30
31      private javax.swing.JButton ivjCancelButton = null;
32      private javax.swing.JPanel ivjJDialogContentPane = null;
33      private javax.swing.JButton ivjOKButton = null;
34      private boolean confirmed;
35      private javax.swing.JLabel ivjIconLabel = null;
36      private javax.swing.JPanel ivjMainPanel = null;
37      private javax.swing.JTextPane ivjMessagePanel = null;
38      private javax.swing.JPanel ivjButtonPanel = null;
39      private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
40
41      /**
42       * ConfirmationBox constructor comment.
43       */
44      public ConfirmationBox() {
45          super();
46          initialize();
47      }
48
49      /**
50       * ConfirmationBox constructor comment.
51       * @param owner java.awt.Dialog
52       */
53      public ConfirmationBox(java.awt.Dialog owner) {
54          super(owner);
55      }
56
57      /**
58       * ConfirmationBox constructor comment.
59       * @param owner java.awt.Dialog
60       * @param title java.lang.String
61       */
62      public ConfirmationBox(java.awt.Dialog owner, String title) {
63          super(owner, title);
64      }
65
66      /**
67       * ConfirmationBox constructor comment.
68       * @param owner java.awt.Dialog
69       * @param title java.lang.String
70       * @param modal boolean
71       */
72      public ConfirmationBox(java.awt.Dialog owner, String title, boolean modal) {
73          super(owner, title, modal);
74      }
75
76      /**
77       * ConfirmationBox constructor comment.
78       */
79  }

```

```

    * @param owner java.awt.Dialog
72    * @param modal boolean
    */
74 public ConfirmationBox(java.awt.Dialog owner, boolean modal) {
    super(owner, modal);
76 }
77 /**
78  * ConfirmationBox constructor comment.
79  * @param owner java.awt.Frame
80  */
80 public ConfirmationBox(java.awt.Frame owner) {
81     super(owner);
82 }
83 /**
84  * ConfirmationBox constructor comment.
85  * @param owner java.awt.Frame
86  * @param title java.lang.String
87  */
87 public ConfirmationBox(java.awt.Frame owner, String title) {
88     super(owner, title);
89 }
90 /**
91  * ConfirmationBox constructor comment.
92  * @param owner java.awt.Frame
93  * @param title java.lang.String
94  * @param modal boolean
95  */
95 public ConfirmationBox(java.awt.Frame owner, String title, boolean modal) {
96     super(owner, title, modal);
97 }
98 /**
99  * ConfirmationBox constructor comment.
100 * @param owner java.awt.Frame
101 * @param modal boolean
102 */
102 public ConfirmationBox(java.awt.Frame owner, boolean modal) {
103     super(owner, modal);
104 }
105 /**
106  * Insert the method's description here.
107  * Creation date: (12-07-00 11:37:04)
108 */
109 public void cancelAndClose()
110 {
111     setConfirmed( false );
112     dispose();
113 }
114 /**
115  * Return the JPanel1 property value.
116  * @return javax.swing.JPanel
117 */
118 /* WARNING: THIS METHOD WILL BE REGENERATED. */
119 private javax.swing.JPanel getButtonPanel() {
120     if (ivjButtonPanel == null) {
121         try {
122             ivjButtonPanel = new javax.swing.JPanel();
123             ivjButtonPanel.setName("ButtonPanel");
124             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
125             getButtonPanel().add(getOKButton(), getOKButton().getName());
126             getButtonPanel().add(getCancelButton(), getCancelButton().getName());
127             // user code begin {1}
128             // user code end
129         } catch (java.lang.Throwable ivjExc) {
130             // user code begin {2}
131             // user code end
132             handleException(ivjExc);
133         }
134     }
135 }

```

```

        }
138     }
139     return ivjButtonPanel;
140 }
141 /**
142  * Return the ButtonPanelFlowLayout property value.
143  * @return java.awt.FlowLayout
144  */
145 /* WARNING: THIS METHOD WILL BE REGENERATED. */
146 private java.awt.FlowLayout getButtonPanelFlowLayout() {
147     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
148     try {
149         /* Create part */
150         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
151         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
152     } catch (java.lang.Throwable ivjExc) {
153         handleException(ivjExc);
154     };
155     return ivjButtonPanelFlowLayout;
156 }
157 /**
158  * Return the CancelButton property value.
159  * @return javax.swing.JButton
160  */
161 /* WARNING: THIS METHOD WILL BE REGENERATED. */
162 private javax.swing.JButton getCancelButton() {
163     if (ivjCancelButton == null) {
164         try {
165             ivjCancelButton = new javax.swing.JButton();
166             ivjCancelButton.setName("CancelButton");
167             ivjCancelButton.setText("Cancel");
168             // user code begin {1}
169             ivjCancelButton.addActionListener( new CancelButtonHandler() );
170             // user code end
171         } catch (java.lang.Throwable ivjExc) {
172             // user code begin {2}
173             // user code end
174             handleException(ivjExc);
175         }
176     }
177     return ivjCancelButton;
178 }
179 /**
180  * Return the IconLabel property value.
181  * @return javax.swing.JLabel
182  */
183 /* WARNING: THIS METHOD WILL BE REGENERATED. */
184 private javax.swing.JLabel getIconLabel() {
185     if (ivjIconLabel == null) {
186         try {
187             ivjIconLabel = new javax.swing.JLabel();
188             ivjIconLabel.setName("IconLabel");
189             ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/
190                 general/ContextualHelp24.gif")));
191             ivjIconLabel.setText("");
192             // user code begin {1}
193             // user code end
194         } catch (java.lang.Throwable ivjExc) {
195             // user code begin {2}
196             // user code end
197             handleException(ivjExc);
198         }
199     }
200     return ivjIconLabel;
201 }
202 /**

```

```

202 * Return the JDialogContentPane property value.
203 * @return javax.swing.JPanel
204 */
205 /* WARNING: THIS METHOD WILL BE REGENERATED. */
206 private javax.swing.JPanel getJDialogContentPane() {
207     if (ivjJDialogContentPane == null) {
208         try {
209             ivjJDialogContentPane = new javax.swing.JPanel();
210             ivjJDialogContentPane.setName("JDialogContentPane");
211             ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
212             getJDialogContentPane().add(getButtonPanel(), "South");
213             getJDialogContentPane().add(getMainPanel(), "Center");
214             // user code begin {1}
215             // user code end
216         } catch (java.lang.Throwable ivjExc) {
217             // user code begin {2}
218             // user code end
219             handleException(ivjExc);
220         }
221     }
222     return ivjJDialogContentPane;
223 }
224 /**
225 * Return the JPanel2 property value.
226 * @return javax.swing.JPanel
227 */
228 /* WARNING: THIS METHOD WILL BE REGENERATED. */
229 private javax.swing.JPanel getMainPanel() {
230     if (ivjMainPanel == null) {
231         try {
232             ivjMainPanel = new javax.swing.JPanel();
233             ivjMainPanel.setName("MainPanel");
234             ivjMainPanel.setLayout(new java.awt.GridBagLayout());
235
236             java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
237             constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
238             constraintsMessagePanel.fill = java.awt.GridBagConstraints.NONE;
239             constraintsMessagePanel.weightx = 1.0;
240             constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
241             getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
242
243             java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
244             constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
245             constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
246             getMainPanel().add(getIconLabel(), constraintsIconLabel);
247             // user code begin {1}
248             getMainPanel().setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ) );
249             // user code end
250         } catch (java.lang.Throwable ivjExc) {
251             // user code begin {2}
252             // user code end
253             handleException(ivjExc);
254         }
255     }
256     return ivjMainPanel;
257 }
258 /**
259 * Return the JTextPanel1 property value.
260 * @return javax.swing.JTextPane
261 */
262 /* WARNING: THIS METHOD WILL BE REGENERATED. */
263 private javax.swing.JTextPane getMessagePanel() {
264     if (ivjMessagePanel == null) {
265         try {
266             ivjMessagePanel = new javax.swing.JTextPane();
267             ivjMessagePanel.setName("MessagePanel");

```

```

268         ivjMessagePanel.setEditable(false);
269         // user code begin {1}
270         ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
271         // user code end
272     } catch (java.lang.Throwable ivjExc) {
273         // user code begin {2}
274         // user code end
275         handleException(ivjExc);
276     }
277 }
278 return ivjMessagePanel;
279 }
280 /**
 * Return the OKButton property value.
281 * @return javax.swing.JButton
 */
284 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private javax.swing.JButton getOKButton() {
286     if (ivjOKButton == null) {
287         try {
288             ivjOKButton = new javax.swing.JButton();
289             ivjOKButton.setName("OKButton");
290             ivjOKButton.setMnemonic('O');
291             ivjOKButton.setText("OK");
292             // user code begin {1}
293             ivjOKButton.addActionListener( new OKButtonHandler() );
294             // user code end
295         } catch (java.lang.Throwable ivjExc) {
296             // user code begin {2}
297             // user code end
298             handleException(ivjExc);
299         }
300     }
301     return ivjOKButton;
302 }
303 /**
304 * Called whenever the part throws an exception.
305 * @param exception java.lang.Throwable
306 */
307 private void handleException(java.lang.Throwable exception) {
308     /* Uncomment the following lines to print uncaught exceptions to stdout */
309     // System.out.println("----- UNCAUGHT EXCEPTION -----");
310     // exception.printStackTrace(System.out);
311 }
312 /**
313 * Initialize the class.
314 */
316 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
318     try {
319         // user code begin {1}
320         // user code end
321         setName("ConfirmationBox");
322         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
323         setSize(436, 181);
324         setEnabled(true);
325         setModal(true);
326         setTitle("Confirmation");
327         setContentPane(getJDialogContentPane());
328     } catch (java.lang.Throwable ivjExc) {
329         handleException(ivjExc);
330     }
331     // user code begin {2}
332     // user code end

```

```

334 }
335 /**
336 * Insert the method's description here.
337 * Creation date: (12-07-00 11:26:11)
338 * @return boolean
339 */
340 public boolean isConfirmed() {
341     return confirmed;
342 }
343 /**
344 * main entrypoint - starts the part when it is run as an application
345 * @param args java.lang.String[]
346 */
347 public static void main(java.lang.String[] args) {
348     try {
349         ConfirmationBox aConfirmationBox;
350         aConfirmationBox = new ConfirmationBox();
351         aConfirmationBox.setModal(true);
352         aConfirmationBox.addWindowListener(new java.awt.event.WindowAdapter() {
353             public void windowClosing(java.awt.event.WindowEvent e) {
354                 System.exit(0);
355             }
356         });
357         aConfirmationBox.setVisible(true);
358     } catch (Throwable exception) {
359         System.err.println("Exception occurred in main() of javax.swing.JDialog");
360         exception.printStackTrace(System.out);
361     }
362 }
363 /**
364 * Insert the method's description here.
365 * Creation date: (12-07-00 11:26:11)
366 * @param newConfirmed boolean
367 */
368 public void setConfirmed(boolean newConfirmed) {
369     confirmed = newConfirmed;
370 }
371 /**
372 * Insert the method's description here.
373 * Creation date: (12-07-00 11:44:14)
374 * @param newText java.lang.String
375 */
376 public void setText(String newText)
377 {
378     getMessagePanel().setText( newText );
379     getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel()
380         .getFontMetrics( getMessagePanel().getFont() ) ) );
381     getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
382     pack();
383     setLocationRelativeTo( getOwner() );
384 }
385 /**
386 * Insert the method's description here.
387 * Creation date: (12-07-00 11:36:53)
388 */
389 public void submitAndClose()
390 {
391     setConfirmed( true );
392     dispose();
393 }

```

Listing C.37: DataPackage.java

```

package peerviewmisc;
2
import java.io.*;

```

```

4 import java.util.Hashtable;
5
6 /**
7 * This class implements a container used for transmitting data between clients.
8 * Creation date: (04-07-00 17:10:18)
9 * @author:
10 */
11
12 public class DataPackage implements Serializable
13 {
14     private final static java.lang.String AUTHOR_NAME = "Author\u00b7name";
15     private final static java.lang.String DOCPATH = "Document\u00b7name\u00b7and\u00b7path";
16     private final static java.lang.String DOCNAME = "Document\u00b7name\u00b7(without\u00b7path)";
17     private final static java.lang.String DESCRIPTION = "Description";
18     private java.util.Properties properties = new java.util.Properties();
19     private java.lang.Object contents = null;
20     private final static java.lang.String WORKGROUP_ID = "Workgroup\u00b7ID";
21     private final static java.lang.String MESSAGE_ID = "Message\u00b7ID";
22     private final static java.lang.String CLIENT_NAME = "Client\u00b7name";
23
24     /**
25      * DataPackage constructor comment.
26      */
27     public DataPackage() {
28         super();
29     }
30
31     /**
32      * Insert the method's description here.
33      * Creation date: (27-10-00 22:45:37)
34      * @return java.lang.String
35      */
36     public final static java.lang.String getAUTHOR_NAME() {
37         return AUTHOR_NAME;
38     }
39
40     /**
41      * Insert the method's description here.
42      * Creation date: (29-10-00 19:05:42)
43      * @return java.lang.String
44      */
45     public final static java.lang.String getClient_NAME() {
46         return CLIENT_NAME;
47     }
48
49     /**
50      * Insert the method's description here.
51      * Creation date: (27-10-00 22:59:23)
52      * @return java.lang.Object
53      */
54     public java.lang.Object getContents() {
55         return contents;
56     }
57
58     /**
59      * Insert the method's description here.
60      * Creation date: (27-10-00 22:52:15)
61      * @return java.lang.String
62      */
63     public final static java.lang.String getDESCRIPTION() {
64         return DESCRIPTION;
65     }
66
67     /**
68      * Insert the method's description here.
69      * Creation date: (27-10-00 22:49:36)
70      * @return java.lang.String
71      */
72     public final static java.lang.String getDOCNAME() {
73         return DOCNAME;
74     }
75

```

```

70 * Insert the method's description here.
71 * Creation date: (27-10-00 22:49:02)
72 * @return java.lang.String
73 */
74 public final static java.lang.String getDOC PATH() {
75     return DOC PATH;
76 }
77 /**
78 * Insert the method's description here.
79 * Creation date: (28-10-00 00:03:49)
80 * @return java.lang.String
81 */
82 public final static java.lang.String getMessage_ID() {
83     return MESSAGE_ID;
84 }
85 /**
86 * Insert the method's description here.
87 * Creation date: (27-10-00 22:55:47)
88 * @return java.util.Properties
89 */
90 public java.util.Properties getProperties() {
91     return properties;
92 }
93 /**
94 * Insert the method's description here.
95 * Creation date: (27-10-00 23:18:45)
96 * @return java.lang.String
97 */
98 public final static java.lang.String getWORKGROUP_ID() {
99     return WORKGROUP_ID;
100 }
101 /**
102 * Insert the method's description here.
103 * Creation date: (27-10-00 22:59:23)
104 * @param newContents java.lang.Object
105 */
106 public void setContents(java.lang.Object newContents) {
107     contents = newContents;
108 }
109 /**
110 * Insert the method's description here.
111 * Creation date: (27-10-00 22:55:47)
112 * @param newProperties java.util.Properties
113 */
114 public void setProperties(java.util.Properties newProperties) {
115     properties = newProperties;
116 }
117 }
```

Listing C.38: DeleteGroup.java

```

package peerviewmisc;
1 /**
2  * This class implements the delete group dialog box.
3  * Creation date: (29-06-00 16:19:12)
4  * @author:
5  */
6 public class DeleteGroup extends DisplayGroup {
7 /**
8  * DeleteGroup constructor comment.
9  */
10 public DeleteGroup() {
11     super();
12     initialize();
13 }
14 /**
15  */
16 }
```

```

    * DeleteGroup constructor comment.
18   * @param owner java.awt.Dialog
    */
20 public DeleteGroup(java.awt.Dialog owner) {
    super(owner);
22 }
/***
24   * DeleteGroup constructor comment.
25   * @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.
33   * @param owner java.awt.Dialog
34   * @param title java.lang.String
35   * @param modal boolean
    */
36 public DeleteGroup(java.awt.Dialog owner, String title, boolean modal) {
38     super(owner, title, modal);
}
40 /**
41   * DeleteGroup constructor comment.
42   * @param owner java.awt.Dialog
43   * @param modal boolean
    */
44 public DeleteGroup(java.awt.Dialog owner, boolean modal) {
46     super(owner, modal);
}
48 /**
49   * DeleteGroup constructor comment.
50   * @param owner java.awt.Frame
    */
52 public DeleteGroup(java.awt.Frame owner) {
    super(owner);
54 }
/***
56   * DeleteGroup constructor comment.
57   * @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.
65   * @param owner java.awt.Frame
66   * @param title java.lang.String
67   * @param modal boolean
    */
68 public DeleteGroup(java.awt.Frame owner, String title, boolean modal) {
70     super(owner, title, modal);
}
72 /**
73   * DeleteGroup constructor comment.
74   * @param owner java.awt.Frame
75   * @param modal boolean
    */
76 public DeleteGroup(java.awt.Frame owner, boolean modal) {
78     super(owner, modal);
}
79 /**
80   * Called whenever the part throws an exception.
81   * @param exception java.lang.Throwable
82 
```

```

        */
84 private void handleException(java.lang.Throwable exception) {
85
86     /* Uncomment the following lines to print uncaught exceptions to stdout */
87     // System.out.println("----- UNCAUGHT EXCEPTION -----");
88     // exception.printStackTrace(System.out);
89 }
90 /**
91 * Initialize the class.
92 */
93 /* WARNING: THIS METHOD WILL BE REGENERATED. */
94 private void initialize() {
95     try {
96         // user code begin {1}
97         // user code end
98         setName("DeleteGroup");
99     } catch (java.lang.Throwable ivjExc) {
100         handleException(ivjExc);
101     }
102     // user code begin {2}
103     setTitle( "DeleteGroup" );
104     editable( false );
105     getOKButton().setText( "Delete" );
106     // user code end
107 }
108 /**
109 * main entrypoint - starts the part when it is run as an application
110 * @param args java.lang.String[]
111 */
112 public static void main(java.lang.String[] args) {
113     try {
114         DeleteGroup aDeleteGroup;
115         aDeleteGroup = new DeleteGroup();
116         aDeleteGroup.setModal(true);
117         aDeleteGroup.addWindowListener(new java.awt.event.WindowAdapter() {
118             public void windowClosing(java.awt.event.WindowEvent e) {
119                 System.exit(0);
120             }
121         });
122         aDeleteGroup.setVisible(true);
123     } catch (Throwable exception) {
124         System.err.println("Exception occurred in main() of serverapp.DeleteGroup");
125         exception.printStackTrace(System.out);
126     }
127 }
128 /**
129 * Insert the method's description here.
130 * Creation date: (27-07-00 09:42:39)
131 */
132 public void submitAndClose() {}

```

Listing C.39: DisplayGroup.java

```

package peerviewmisc;
1
2     import javax.swing.JComponent;
3     import javax.swing.JTextField;
4     import javax.swing.text.JTextComponent;
5     import java.awt.Component;
6     import javax.swing.event.*;
7     import java.awt.event.*;
8
9
10    /**
11     * This abstract class is the root of the class hierarchy of dialog boxes that accept input from the
12     * user.
13     */
14    * Creation date: (29-06-00 10:21:35)

```

```

    * @author:
14   */
15  public abstract class DisplayGroup extends javax.swing.JDialog {
16
17      public class OKButtonHandler implements ActionListener
18      {
19          public void actionPerformed( ActionEvent ae )
20          {
21              submitAndClose();
22          }
23      }
24
25      public class CancelButtonHandler implements ActionListener
26      {
27          public void actionPerformed( ActionEvent ae )
28          {
29              cancelAndClose();
30          }
31      }
32      protected javax.swing.JButton ivjCancelButton = null;
33      protected javax.swing.JLabel ivjDescriptionLabel = null;
34      protected javax.swing.JLabel ivjGroupNameLabel = null;
35      protected javax.swing.JPanel ivjJDialogContentPane = null;
36      protected javax.swing.JButton ivjOKButton = null;
37      protected javax.swing.JPanel ivjButtonPanel = null;
38      protected java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
39      protected javax.swing.JPanel ivjMainPanel = null;
40      protected WorkGroup workGroup = null;
41      protected javax.swing.JTextPane ivjDescriptionField = null;
42      protected javax.swing.JTextField ivjGroupNameField = null;
43      protected clientapp.GroupDirectory groupDirectory;
44      protected boolean isDiscarded;
45      protected javax.swing.JScrollPane ivjDescriptionScrollPane = null;
46 /**
47  * DisplayGroup constructor comment.
48 */
49  public DisplayGroup() {
50      super();
51      initialize();
52 }
53 /**
54  * DisplayGroup constructor comment.
55  * @param owner java.awt.Dialog
56 */
57  public DisplayGroup(java.awt.Dialog owner) {
58      super(owner);
59 }
60 /**
61  * DisplayGroup constructor comment.
62  * @param owner java.awt.Dialog
63  * @param title java.lang.String
64 */
65  public DisplayGroup(java.awt.Dialog owner, String title) {
66      super(owner, title);
67 }
68 /**
69  * DisplayGroup constructor comment.
70  * @param owner java.awt.Dialog
71  * @param title java.lang.String
72  * @param modal boolean
73 */
74  public DisplayGroup(java.awt.Dialog owner, String title, boolean modal) {
75      super(owner, title, modal);
76 }
77 /**
78  * DisplayGroup constructor comment.

```

```

    * @param owner java.awt.Dialog
80   * @param modal boolean
81   */
82 public DisplayGroup(java.awt.Dialog owner, boolean modal) {
83     super(owner, modal);
84 }
85 /**
86  * DisplayGroup constructor comment.
87  * @param owner java.awt.Frame
88  */
89 public DisplayGroup(java.awt.Frame owner) {
90   super(owner);
91 }
92 /**
93  * DisplayGroup constructor comment.
94  * @param owner java.awt.Frame
95  * @param title java.lang.String
96  */
97 public DisplayGroup(java.awt.Frame owner, String title) {
98   super(owner, title);
99 }
100 /**
101  * DisplayGroup constructor comment.
102  * @param owner java.awt.Frame
103  * @param title java.lang.String
104  * @param modal boolean
105  */
106 public DisplayGroup(java.awt.Frame owner, String title, boolean modal) {
107   super(owner, title, modal);
108 }
109 /**
110  * DisplayGroup constructor comment.
111  * @param owner java.awt.Frame
112  * @param modal boolean
113  */
114 public DisplayGroup(java.awt.Frame owner, boolean modal) {
115   super(owner, modal);
116 }
117 /**
118  * Insert the method's description here.
119  * Creation date: (26-07-00 19:28:46)
120  */
121 public void cancelAndClose()
122 {
123   setIsDiscarded( true );
124   dispose();
125 }
126 /**
127  * Set editability of text fields in the main panel of the display group dialog box.
128  * Creation date: (29-06-00 15:26:57)
129  * @param edit boolean
130  */
131 public void editable(boolean edit)
132 {
133   Component components[] = getMainPanel().getComponents();
134   for (int i = 0; i < components.length; i++)
135   {
136     if (components[i] instanceof JTextComponent)
137     {
138       ((JTextComponent) components[i]).setEditable(edit);
139     }
140   }
141 }
142 /**
143  * Return the JPanel2 property value.

```

```

    * @return javax.swing.JPanel
146   */
147   /* WARNING: THIS METHOD WILL BE REGENERATED. */
148   private javax.swing.JPanel getButtonPanel() {
149     if (ivjButtonPanel == null) {
150       try {
151         ivjButtonPanel = new javax.swing.JPanel();
152         ivjButtonPanel.setName("ButtonPanel");
153         ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
154         getButtonPanel().add(getOKButton(), getOKButton().getName());
155         getButtonPanel().add(getCancelButton(), getCancelButton().getName());
156         // user code begin {1}
157         // user code end
158       } catch (java.lang.Throwable ivjExc) {
159         // user code begin {2}
160         // user code end
161         handleException(ivjExc);
162       }
163     }
164     return ivjButtonPanel;
165   }
166 /**
167  * Return the ButtonPanelFlowLayout property value.
168  * @return java.awt.FlowLayout
169  */
170 /* WARNING: THIS METHOD WILL BE REGENERATED. */
171 private java.awt.FlowLayout getButtonPanelFlowLayout() {
172   java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
173   try {
174     /* Create part */
175     ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
176     ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
177   } catch (java.lang.Throwable ivjExc) {
178     handleException(ivjExc);
179   };
180   return ivjButtonPanelFlowLayout;
181 }
182 /**
183  * Return the CancelButton property value.
184  * @return javax.swing.JButton
185  */
186 /* WARNING: THIS METHOD WILL BE REGENERATED. */
187 public javax.swing.JButton getCancelButton() {
188   if (ivjCancelButton == null) {
189     try {
190       ivjCancelButton = new javax.swing.JButton();
191       ivjCancelButton.setName("CancelButton");
192       ivjCancelButton.setText("Cancel");
193       // user code begin {1}
194       ivjCancelButton.addActionListener( new CancelButtonHandler() );
195       // user code end
196     } catch (java.lang.Throwable ivjExc) {
197       // user code begin {2}
198       // user code end
199       handleException(ivjExc);
200     }
201   }
202   return ivjCancelButton;
203 }
204 /**
205  * Return the JPanel property value.
206  * @return javax.swing.JPanel
207  */
208 /* WARNING: THIS METHOD WILL BE REGENERATED. */
209 private javax.swing.JPanel getDescriptionField() {
210   if (ivjDescriptionField == null) {

```

```

try {
212     ivjDescriptionField = new javax.swing.JTextPane();
213     ivjDescriptionField.setName("DescriptionField");
214     ivjDescriptionField.setBounds(0, 0, 312, 137);
215     // user code begin {1}
216     // user code end
217 } catch (java.lang.Throwable ivjExc) {
218     // user code begin {2}
219     // user code end
220     handleException(ivjExc);
221 }
222 }
223 return ivjDescriptionField;
224 }
225 /**
226 * Return the DescriptionLabel property value.
227 * @return javax.swing.JLabel
228 */
229 /* WARNING: THIS METHOD WILL BE REGENERATED. */
230 private javax.swing.JLabel getDescriptionLabel() {
231     if (ivjDescriptionLabel == null) {
232         try {
233             ivjDescriptionLabel = new javax.swing.JLabel();
234             ivjDescriptionLabel.setName("DescriptionLabel");
235             ivjDescriptionLabel.setText("Description");
236             // user code begin {1}
237             // user code end
238         } catch (java.lang.Throwable ivjExc) {
239             // user code begin {2}
240             // user code end
241             handleException(ivjExc);
242         }
243     }
244     return ivjDescriptionLabel;
245 }
246 /**
247 * Return the DescriptionScrollPane property value.
248 * @return javax.swing.JScrollPane
249 */
250 /* WARNING: THIS METHOD WILL BE REGENERATED. */
251 private javax.swing.JScrollPane getDescriptionScrollPane() {
252     if (ivjDescriptionScrollPane == null) {
253         try {
254             ivjDescriptionScrollPane = new javax.swing.JScrollPane();
255             ivjDescriptionScrollPane.setName("DescriptionScrollPane");
256             getDescriptionScrollPane().setViewportView(getDescriptionField());
257             // user code begin {1}
258             // user code end
259         } catch (java.lang.Throwable ivjExc) {
260             // user code begin {2}
261             // user code end
262             handleException(ivjExc);
263         }
264     }
265     return ivjDescriptionScrollPane;
266 }
267 /**
268 * Insert the method's description here.
269 * Creation date: (26-07-00 17:28:18)
270 * @return clientapp.GroupDirectory
271 */
272 public clientapp.GroupDirectory getGroupDirectory() {
273     return groupDirectory;
274 }
275 /**
276 * Return the JTextField1 property value.

```

```

278     * @return javax.swing.JTextField
279     */
280     /* WARNING: THIS METHOD WILL BE REGENERATED. */
281     private javax.swing.JTextField getGroupNameField() {
282         if (ivjGroupNameField == null) {
283             try {
284                 ivjGroupNameField = new javax.swing.JTextField();
285                 ivjGroupNameField.setName("GroupNameField");
286                 // user code begin {1}
287                 // user code end
288                 } catch (java.lang.Throwable ivjExc) {
289                     // user code begin {2}
290                     // user code end
291                     handleException(ivjExc);
292                 }
293             }
294         return ivjGroupNameField;
295     }
296     /**
297      * Return the GroupNameLabel property value.
298      * @return javax.swing.JLabel
299      */
300     /* WARNING: THIS METHOD WILL BE REGENERATED. */
301     private javax.swing.JLabel getGroupNameLabel() {
302         if (ivjGroupNameLabel == null) {
303             try {
304                 ivjGroupNameLabel = new javax.swing.JLabel();
305                 ivjGroupNameLabel.setName("GroupNameLabel");
306                 ivjGroupNameLabel.setText("Group\u00d7name");
307                 // user code begin {1}
308                 // user code end
309                 } catch (java.lang.Throwable ivjExc) {
310                     // user code begin {2}
311                     // user code end
312                     handleException(ivjExc);
313                 }
314             }
315         return ivjGroupNameLabel;
316     }
317     /**
318      * Return the JDialoContentPane property value.
319      * @return javax.swing.JPanel
320      */
321     /* WARNING: THIS METHOD WILL BE REGENERATED. */
322     private javax.swing.JPanel getJDialoContentPane() {
323         if (ivjJDialoContentPane == null) {
324             try {
325                 ivjJDialoContentPane = new javax.swing.JPanel();
326                 ivjJDialoContentPane.setName("JDialoContentPane");
327                 ivjJDialoContentPane.setLayout(new java.awt.BorderLayout());
328                 getJDialoContentPane().add(getMainPanel(), "Center");
329                 getJDialoContentPane().add(getButtonPanel(), "South");
330                 // user code begin {1}
331                 // user code end
332                 } catch (java.lang.Throwable ivjExc) {
333                     // user code begin {2}
334                     // user code end
335                     handleException(ivjExc);
336                 }
337             }
338         return ivjJDialoContentPane;
339     }
340     /**
341      * Return the JPanel1 property value.
342      * @return javax.swing.JPanel
343      */

```

```

/* 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.
391 * @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}
        // user code end
        handleException(ivjExc);
    }

```

```

408         }
409     }
410     return ivjOKButton;
411 }
412 /**
413  * Insert the method's description here.
414  * Creation date: (03-07-00 20:56:36)
415  * @return peerviewmisc.WorkGroup
416 */
417 public WorkGroup getWorkGroup() {
418     return workGroup;
419 }
420 /**
421  * Called whenever the part throws an exception.
422  * @param exception java.lang.Throwable
423 */
424 private void handleException(java.lang.Throwable exception) {
425
426     /* Uncomment the following lines to print uncaught exceptions to stdout */
427     // System.out.println("----- UNCAUGHT EXCEPTION -----");
428     // exception.printStackTrace(System.out);
429 }
430 /**
431  * Initialize the class.
432 */
433 /* WARNING: THIS METHOD WILL BE REGENERATED. */
434 private void initialize() {
435     try {
436         // user code begin {1}
437         // user code end
438         setName("DisplayGroup");
439         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
440         setSize(426, 240);
441         setContentPane(getJDialogContentPane());
442     } catch (java.lang.Throwable ivjExc) {
443         handleException(ivjExc);
444     }
445     // user code begin {2}
446     setModal( true );
447     // set to true as default to ensure that cancelling using the upper right corner X doesn't result in
448     // a spurious entry
449     setIsDiscarded( true );
450     setLocationRelativeTo( getOwner() );
451
452     // user code end
453 }
454 /**
455  * Insert the method's description here.
456  * Creation date: (26-07-00 19:10:38)
457  * @return boolean
458 */
459 public boolean isIsDiscarded() {
460     return isDiscarded;
461 }
462 /**
463  * main entrypoint - starts the part when it is run as an application
464  * @param args java.lang.String[]
465 */
466 public static void main(java.lang.String[] args) {
467     /* try {
468         DisplayGroup aDisplayGroup;
469         aDisplayGroup = new DisplayGroup();
470         aDisplayGroup.setModal(true);
471         aDisplayGroup.addWindowListener(new java.awt.event.WindowAdapter() {
472             public void windowClosing(java.awt.event.WindowEvent e) {
473                 System.exit(0);
474             }
475         });
476     }*/
477 }

```

```

        };
474     });
475     aDisplayGroup.setVisible(true);
476 } catch (Throwable exception) {
477     System.err.println("Exception occurred in main() of javax.swing.JDialog");
478     exception.printStackTrace(System.out);
479 }
480 */
481 }
482 /**
483 * Insert the method's description here.
484 * Creation date: (26-07-00 17:28:18)
485 * @param newGroupDirectory clientapp.GroupDirectory
486 */
487 public void setGroupDirectory(clientapp.GroupDirectory newGroupDirectory) {
488     groupDirectory = newGroupDirectory;
489 }
490 /**
491 * Insert the method's description here.
492 * Creation date: (26-07-00 19:10:38)
493 * @param newIsDiscarded boolean
494 */
495 public void setIsDiscarded(boolean newIsDiscarded) {
496     isDiscarded = newIsDiscarded;
497 }
498 /**
499 * Insert the method's description here.
500 * Creation date: (03-07-00 20:56:36)
501 * @param newWorkGroup peerviewmisc.WorkGroup
502 */
503 public void setWorkGroup(WorkGroup newWorkGroup)
504 {
505     workGroup = newWorkGroup;
506     getGroupNameField().setText( workGroup.getGroupName() );
507     getDescriptionField().setText( workGroup.getDescription() );
508 }
509 /**
510 * Insert the method's description here.
511 * Creation date: (27-07-00 00:02:29)
512 */
513 public abstract void submitAndClose();
514 }

```

Listing C.40: DocPackage.java

```

package peerviewmisc;
2
3 /**
4  * This class implements a container which represents a node in the client panorama by storing the name
4  * of its associated
5  * document and the size, position and locking status of the node.
6  * Creation date: (11-10-00 12:03:22)
7  * @author:
8 */
9 public class DocPackage implements java.io.Serializable {
10     private java.lang.String docNameAndPath = null;
11     private java.awt.Dimension size = null;
12     private java.awt.Dimension position = null;
13     private java.lang.String NodeLocked = null;
14 /**
15  * DocPackage constructor comment.
16 */
17     public DocPackage() {
18         super();
19     }
20 /**

```

```

    * Insert the method's description here.
22   * Creation date: (11-10-00 12:12:19)
    * @param docNameAndPathArg java.lang.String
24   * @param size java.awt.Dimension
    * @param position java.awt.Dimension
26   */
27
28   public DocPackage(String docNameAndPathArg, java.awt.Dimension sizeArg, java.awt.Dimension positionArg )
29   {
30       setDocNameAndPath( docNameAndPathArg );
31       setSize( sizeArg );
32       setPosition( positionArg );
33   }
34
35   /**
36   * Insert the method's description here.
37   * Creation date: (11-10-00 12:03:36)
38   * @return java.lang.String
39   */
40
41   public java.lang.String getDocNameAndPath() {
42       return docNameAndPath;
43   }
44
45   /**
46   * Insert the method's description here.
47   * Creation date: (16-10-00 09:51:21)
48   * @return java.lang.String
49   */
50
51   public java.lang.String getNodeLocked() {
52       return NodeLocked;
53   }
54
55   /**
56   * Insert the method's description here.
57   * Creation date: (11-10-00 12:04:44)
58   * @return java.awt.Dimension
59   */
60
61   public java.awt.Dimension getPosition() {
62       return position;
63   }
64
65   /**
66   * Insert the method's description here.
67   * Creation date: (11-10-00 12:03:54)
68   * @return java.awt.Dimension
69   */
70
71   public java.awt.Dimension getSize() {
72       return size;
73   }
74
75   /**
76   * Insert the method's description here.
77   * Creation date: (11-10-00 12:03:36)
78   * @param newDocNameAndPath java.lang.String
79   */
80
81   public void setDocNameAndPath(java.lang.String newDocNameAndPath) {
82       docNameAndPath = newDocNameAndPath;
83   }
84
85   /**
86   * Insert the method's description here.
87   * Creation date: (16-10-00 09:51:21)
88   * @param newNodeLocked java.lang.String
89   */
90
91   public void setNodeLocked(java.lang.String newNodeLocked) {
92       NodeLocked = newNodeLocked;
93   }
94
95   /**
96   * Insert the method's description here.
97   * Creation date: (11-10-00 12:04:44)
98   * @param newPosition java.awt.Dimension
99   */
100
101  public void setPosition(java.awt.Dimension newPosition) {

```

```

        position = newPosition;
88    }
89    /**
90     * Insert the method's description here.
91     * Creation date: (11-10-00 12:03:54)
92     * @param newSize java.awt.Dimension
93     */
94    public void setSize(java.awt.Dimension newSize) {
95        size = newSize;
96    }
97

```

Listing C.41: EditGroup.java

```

package peerviewmisc;
2
3    /**
4     * The edit group dialog box.
5     * Creation date: (29-06-00 16:08:16)
6     * @author:
7     */
8    public class EditGroup extends DisplayGroup
{
10
11    /**
12     * EditGroup constructor comment.
13     */
14    public EditGroup() {
15        super();
16        initialize();
17    }
18    /**
19     * EditGroup constructor comment.
20     * @param owner java.awt.Dialog
21     */
22    public EditGroup(java.awt.Dialog owner) {
23        super(owner);
24    }
25    /**
26     * EditGroup constructor comment.
27     * @param owner java.awt.Dialog
28     * @param title java.lang.String
29     */
30    public EditGroup(java.awt.Dialog owner, String title) {
31        super(owner, title);
32    }
33    /**
34     * EditGroup constructor comment.
35     * @param owner java.awt.Dialog
36     * @param title java.lang.String
37     * @param modal boolean
38     */
39    public EditGroup(java.awt.Dialog owner, String title, boolean modal) {
40        super(owner, title, modal);
41    }
42    /**
43     * EditGroup constructor comment.
44     * @param owner java.awt.Dialog
45     * @param modal boolean
46     */
47    public EditGroup(java.awt.Dialog owner, boolean modal) {
48        super(owner, modal);
49    }
50    /**
51     * EditGroup constructor comment.
52     * @param owner java.awt.Frame
53     */

```

```

54 public EditGroup(java.awt.Frame owner) {
55     super(owner);
56 }
57 /**
58 * EditGroup constructor comment.
59 * @param owner java.awt.Frame
60 * @param title java.lang.String
61 */
62 public EditGroup(java.awt.Frame owner, String title) {
63     super(owner, title);
64 }
65 /**
66 * EditGroup constructor comment.
67 * @param owner java.awt.Frame
68 * @param title java.lang.String
69 * @param modal boolean
70 */
71 public EditGroup(java.awt.Frame owner, String title, boolean modal) {
72     super(owner, title, modal);
73 }
74 /**
75 * EditGroup constructor comment.
76 * @param owner java.awt.Frame
77 * @param modal boolean
78 */
79 public EditGroup(java.awt.Frame owner, boolean modal) {
80     super(owner, modal);
81 }
82 /**
83 * Called whenever the part throws an exception.
84 * @param exception java.lang.Throwable
85 */
86 private void handleException(java.lang.Throwable exception) {
87
88     /* Uncomment the following lines to print uncaught exceptions to stdout */
89     // System.out.println("----- UNCAUGHT EXCEPTION -----");
90     // exception.printStackTrace(System.out);
91 }
92 /**
93 * Initialize the class.
94 */
95 /* WARNING: THIS METHOD WILL BE REGENERATED. */
96 private void initialize() {
97     try {
98         // user code begin {1}
99         // user code end
100     setName("EditGroup");
101     } catch (java.lang.Throwable ivjExc) {
102         handleException(ivjExc);
103     }
104     // user code begin {2}
105     setTitle( "EditGroup" );
106     editable( true );
107     getOKButton().setText( "Submit" );
108     getOKButton().setMnemonic( 's' );
109     getCancelButton().setText( "Discard" );
110     getCancelButton().setMnemonic( 'd' );
111     // user code end
112 }
113 /**
114 * main entrypoint - starts the part when it is run as an application
115 * @param args java.lang.String[]
116 */
117 public static void main(java.lang.String[] args) {
118     try {
119         EditGroup aEditGroup;

```

```

120     aEditGroup = new EditGroup();
121     aEditGroup.setModal(true);
122     aEditGroup.addWindowListener(new java.awt.event.WindowAdapter() {
123         public void windowClosing(java.awt.event.WindowEvent e) {
124             System.exit(0);
125         };
126     });
127     aEditGroup.setVisible(true);
128 } catch (Throwable exception) {
129     System.err.println("Exception occurred in main() of serverapp.EditGroup");
130     exception.printStackTrace(System.out);
131 }
132 */
133 * Insert the method's description here.
134 * Creation date: (04-07-00 00:15:52)
135 */
136 public void submitAndClose()
137 {
138     String groupName = ivjGroupNameField.getText();
139     String description = ivjDescriptionField.getText();
140     ErrorBox errorBox = new ErrorBox();
141
142     if (groupName.length() == 0 )
143     {
144         errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
145         errorBox.show();
146         return;
147     }
148     getWorkGroup().setDescription( ivjDescriptionField.getText() );
149     getWorkGroup().setGroupName( ivjGroupNameField.getText() );
150     setIsDiscarded( false );
151     dispose();
152 }
153
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
11 /**
12  * The dialog box displayed when the user is notified of an error.
13  * Creation date: (07-06-00 00:37:01)
14  * @author:
15  */
16 public class ErrorBox extends JDialog {
17
18     public class CloseButtonHandler implements ActionListener
19     {
20         public void actionPerformed( ActionEvent ae )
21         {
22             dispose();
23         }
24     }
25     private JPanel ivjJDialogContentPane = null;
26     private JPanel ivjButtonPanel = null;
27     private FlowLayout ivjButtonPanelFlowLayout = null;
28     private JLabel ivjIconLabel = null;
29     private JPanel ivjMainPanel = null;

```

```

30     private JTextPane ivjMessagePanel = null;
31     private JButton ivjCloseButton = null;
32 /**
33  * ErrorBox constructor comment.
34 */
35 public ErrorBox() {
36     super();
37     initialize();
38 }
39 /**
40  * ErrorBox constructor comment.
41  * @param owner java.awt.Dialog
42 */
43 public ErrorBox(java.awt.Dialog owner) {
44     super(owner);
45 }
46 /**
47  * ErrorBox constructor comment.
48  * @param owner java.awt.Dialog
49  * @param title java.lang.String
50 */
51 public ErrorBox(java.awt.Dialog owner, String title) {
52     super(owner, title);
53 }
54 /**
55  * ErrorBox constructor comment.
56  * @param owner java.awt.Dialog
57  * @param title java.lang.String
58  * @param modal boolean
59 */
60 public ErrorBox(java.awt.Dialog owner, String title, boolean modal) {
61     super(owner, title, modal);
62 }
63 /**
64  * ErrorBox constructor comment.
65  * @param owner java.awt.Dialog
66  * @param modal boolean
67 */
68 public ErrorBox(java.awt.Dialog owner, boolean modal) {
69     super(owner, modal);
70 }
71 /**
72  * ErrorBox constructor comment.
73  * @param owner java.awt.Frame
74 */
75 public ErrorBox(java.awt.Frame owner) {
76     super(owner);
77 }
78 /**
79  * ErrorBox constructor comment.
80  * @param owner java.awt.Frame
81  * @param title java.lang.String
82 */
83 public ErrorBox(java.awt.Frame owner, String title) {
84     super(owner, title);
85 }
86 /**
87  * ErrorBox constructor comment.
88  * @param owner java.awt.Frame
89  * @param title java.lang.String
90  * @param modal boolean
91 */
92 public ErrorBox(java.awt.Frame owner, String title, boolean modal) {
93     super(owner, title, modal);
94 }
95 /**

```

```

96  * ErrorBox constructor comment.
97  * @param owner java.awt.Frame
98  * @param modal boolean
99  */
100 public ErrorBox(java.awt.Frame owner, boolean modal) {
101     super(owner, modal);
102 }
103 /**
104  * Return the JPanel1 property value.
105  * @return javax.swing.JPanel
106  */
107 /* WARNING: THIS METHOD WILL BE REGENERATED. */
108 private javax.swing.JPanel getButtonPanel() {
109     if (ivjButtonPanel == null) {
110         try {
111             ivjButtonPanel = new javax.swing.JPanel();
112             ivjButtonPanel.setName("ButtonPanel");
113             ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
114             getButtonPanel().add(getCloseButton(), getCloseButton().getName());
115             // user code begin {1}
116             // user code end
117             } catch (java.lang.Throwable ivjExc) {
118                 // user code begin {2}
119                 // user code end
120                 handleException(ivjExc);
121             }
122         }
123         return ivjButtonPanel;
124     }
125 /**
126  * Return the ButtonPanelFlowLayout property value.
127  * @return java.awt.FlowLayout
128  */
129 /* WARNING: THIS METHOD WILL BE REGENERATED. */
130 private java.awt.FlowLayout getButtonPanelFlowLayout() {
131     java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
132     try {
133         /* Create part */
134         ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
135         ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
136     } catch (java.lang.Throwable ivjExc) {
137         handleException(ivjExc);
138     };
139     return ivjButtonPanelFlowLayout;
140 }
141 /**
142  * Return the OK property value.
143  * @return javax.swing.JButton
144  */
145 /* WARNING: THIS METHOD WILL BE REGENERATED. */
146 private javax.swing.JButton getCloseButton() {
147     if (ivjCloseButton == null) {
148         try {
149             ivjCloseButton = new javax.swing.JButton();
150             ivjCloseButton.setName("CloseButton");
151             ivjCloseButton.setMnemonic('C');
152             ivjCloseButton.setText("Close");
153             // user code begin {1}
154             ivjCloseButton.addActionListener( new CloseButtonHandler() );
155             // user code end
156         } catch (java.lang.Throwable ivjExc) {
157             // user code begin {2}
158             // user code end
159             handleException(ivjExc);
160         }
161     }

```

```

162     return ivjCloseButton;
163 }
164 /**
165  * Return the IconLabel property value.
166  * @return javax.swing.JLabel
167  */
168 /* WARNING: THIS METHOD WILL BE REGENERATED. */
169 private javax.swing.JLabel getIconLabel() {
170     if (ivjIconLabel == null) {
171         try {
172             ivjIconLabel = new javax.swing.JLabel();
173             ivjIconLabel.setName("IconLabel");
174             ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Stop24.gif")));
175             ivjIconLabel.setText("");
176             // user code begin {1}
177             // user code end
178         } catch (java.lang.Throwable ivjExc) {
179             // user code begin {2}
180             // user code end
181             handleException(ivjExc);
182         }
183     }
184     return ivjIconLabel;
185 }
186 /**
187  * Return the JDialgContentPane property value.
188  * @return javax.swing.JPanel
189  */
190 /* WARNING: THIS METHOD WILL BE REGENERATED. */
191 private javax.swing.JPanel getJDialgContentPane() {
192     if (ivjJDialgContentPane == null) {
193         try {
194             ivjJDialgContentPane = new javax.swing.JPanel();
195             ivjJDialgContentPane.setName("JDialgContentPane");
196             ivjJDialgContentPane.setLayout(new java.awt.BorderLayout());
197             getJDialgContentPane().add(getButtonPanel(), "South");
198             getJDialgContentPane().add(getMainPanel(), "Center");
199             // user code begin {1}
200             // user code end
201         } catch (java.lang.Throwable ivjExc) {
202             // user code begin {2}
203             // user code end
204             handleException(ivjExc);
205         }
206     }
207     return ivjJDialgContentPane;
208 }
209 /**
210  * Return the JPanel1 property value.
211  * @return javax.swing.JPanel
212  */
213 /* WARNING: THIS METHOD WILL BE REGENERATED. */
214 private javax.swing.JPanel getMainPanel() {
215     if (ivjMainPanel == null) {
216         try {
217             ivjMainPanel = new javax.swing.JPanel();
218             ivjMainPanel.setName("MainPanel");
219             ivjMainPanel.setLayout(new java.awt.GridBagLayout());
220
221             java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
222             constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
223             constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
224             constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
225             getMainPanel().add(getIconLabel(), constraintsIconLabel);
226

```

```

228         java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
229         constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
230         constraintsMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
231         constraintsMessagePanel.weightx = 1.0;
232         constraintsMessagePanel.weighty = 1.0;
233         constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
234         getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
235         // user code begin {1}
236         ivjMainPanel.setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ) );
237         // user code end
238     } catch (java.lang.Throwable ivjExc) {
239         // user code begin {2}
240         // user code end
241         handleException(ivjExc);
242     }
243 }
244 /**
245 * Return the JTextPanel1 property value.
246 * @return javax.swing.JTextPane
247 */
248 /* WARNING: THIS METHOD WILL BE REGENERATED. */
249 private javax.swing.JTextPane getMessagePanel() {
250     if (ivjMessagePanel == null) {
251         try {
252             ivjMessagePanel = new javax.swing.JTextPane();
253             ivjMessagePanel.setName("MessagePanel");
254             ivjMessagePanel.setFont(new java.awt.Font("sansserif.bold", 1, 12));
255             ivjMessagePanel.setBackground(java.awt.Color.lightGray);
256             ivjMessagePanel.setEditable(false);
257             // user code begin {1}
258             ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
259             // user code end
260         } catch (java.lang.Throwable ivjExc) {
261             // user code begin {2}
262             // user code end
263             handleException(ivjExc);
264         }
265     }
266     return ivjMessagePanel;
267 }
268 /**
269 * Called whenever the part throws an exception.
270 * @param exception java.lang.Throwable
271 */
272 private void handleException(java.lang.Throwable exception) {
273     /* Uncomment the following lines to print uncaught exceptions to stdout */
274     // System.out.println("----- UNCAUGHT EXCEPTION -----");
275     // exception.printStackTrace(System.out);
276 }
277 /**
278 * Initialize the class.
279 */
280 /* WARNING: THIS METHOD WILL BE REGENERATED. */
281 private void initialize() {
282     try {
283         // user code begin {1}
284         // user code end
285         setName("ErrorBox");
286         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
287         setSize(401, 191);
288         setModal(true);
289         setTitle("ErrorBox");
290         setContentPane(getJDialogContentPane());
291     }

```

```

294     } catch (java.lang.Throwable ivjExc) {
295         handleException(ivjExc);
296     }
297     // user code begin {2}
298     // user code end
299 }
300 /**
301 * main entrypoint - starts the part when it is run as an application
302 * @param args java.lang.String[]
303 */
304 public static void main(java.lang.String[] args) {
305     try {
306         ErrorBox aErrorBox;
307         aErrorBox = new ErrorBox();
308         aErrorBox.setModal(true);
309         aErrorBox.addWindowListener(new java.awt.event.WindowAdapter() {
310             public void windowClosing(java.awt.event.WindowEvent e) {
311                 System.exit(0);
312             }
313         });
314         aErrorBox.setText("ASFADSRSRFDSDFADSFDSFADFDASFSASDFSDFASFSFADDSFASDFSADFSFADFDASFDSFDSFASDFS");
315         aErrorBox.setVisible(true);
316     } catch (Throwable exception) {
317         System.err.println("Exception occurred in main() of javax.swing.JDialog");
318         exception.printStackTrace(System.out);
319     }
320 }
321 /**
322 * Insert the method's description here.
323 * Creation date: (10-07-00 22:52:54)
324 * @param newText java.lang.String
325 */
326 public void setText(String newText)
327 {
328     getMessagePanel().setText( newText );
329     getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel()
330         .getFontMetrics( getMessagePanel().getFont() ) ) );
331     getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
332     pack();
333     setLocationRelativeTo( getOwner() );
334 }
335 }

```

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
7   * implementation of the PeerView
8  * help system.
9  * Creation date: (29-05-00 12:39:46)
10 * @author:
11 */
12 public class HelpDialog extends javax.swing.JDialog {
13     private javax.swing.JPanel ivjHelpDialog = null;
14     private javax.swing.JPanel ivj JPanel1 = null;
15     private java.awt.FlowLayout ivj JPanel1FlowLayout = null;
16     private javax.swing.JSplitPane ivj JSplitPane1 = null;
17     private javax.swing.JTextPane ivj JTextPane1 = null;
18     private javax.swing.JButton ivjOKButton = null;
19     IvjEventHandler ivjEventHandler = new IvjEventHandler();

```

```

20 class IvjEventHandler implements java.awt.event.ActionListener {
21     public void actionPerformed(java.awt.event.ActionEvent e) {
22         if (e.getSource() == HelpDialog.this.getOKButton())
23             connEtoM1(e);
24     }
25 };
26 private javax.swing.JTree ivjTopicTree = null;
27 /**
28 * HelpDialog constructor comment.
29 */
30 public HelpDialog() {
31     super();
32     initialize();
33 }
34 /**
35 * HelpDialog constructor comment.
36 * @param owner java.awt.Dialog
37 */
38 public HelpDialog(java.awt.Dialog owner) {
39     super(owner);
40 }
41 /**
42 * HelpDialog constructor comment.
43 * @param owner java.awt.Dialog
44 * @param title java.lang.String
45 */
46 public HelpDialog(java.awt.Dialog owner, String title) {
47     super(owner, title);
48 }
49 /**
50 * HelpDialog constructor comment.
51 * @param owner java.awt.Dialog
52 * @param title java.lang.String
53 * @param modal boolean
54 */
55 public HelpDialog(java.awt.Dialog owner, String title, boolean modal) {
56     super(owner, title, modal);
57 }
58 /**
59 * HelpDialog constructor comment.
60 * @param owner java.awt.Dialog
61 * @param modal boolean
62 */
63 public HelpDialog(java.awt.Dialog owner, boolean modal) {
64     super(owner, modal);
65 }
66 /**
67 * HelpDialog constructor comment.
68 * @param owner java.awt.Frame
69 */
70 public HelpDialog(java.awt.Frame owner) {
71     super(owner);
72 }
73 /**
74 * HelpDialog constructor comment.
75 * @param owner java.awt.Frame
76 * @param title java.lang.String
77 */
78 public HelpDialog(java.awt.Frame owner, String title) {
79     super(owner, title);
80 }
81 /**
82 * HelpDialog constructor comment.
83 * @param owner java.awt.Frame
84 * @param title java.lang.String
85 * @param modal boolean

```

```

86  /*
87   * HelpDialog constructor comment.
88   * @param owner java.awt.Frame
89   * @param modal boolean
90   */
91  public HelpDialog(java.awt.Frame owner, String title, boolean modal) {
92      super(owner, title, modal);
93  }
94  /**
95   * HelpDialog constructor comment.
96   * @param owner java.awt.Frame
97   * @param modal boolean
98   */
99  public HelpDialog(java.awt.Frame owner, boolean modal) {
100     super(owner, modal);
101 }
102 /**
103  * connEtoM1: (OKButton.action.actionPerformed(java.awt.event.ActionEvent) --> HelpDialog.dispose()V)
104  * @param arg1 java.awt.event.ActionEvent
105  */
106 /* WARNING: THIS METHOD WILL BE REGENERATED. */
107 private void connEtoM1(java.awt.event.ActionEvent arg1) {
108     try {
109         // user code begin {1}
110         // user code end
111         this.dispose();
112         // user code begin {2}
113         // user code end
114     } catch (java.lang.Throwable ivjExc) {
115         // user code begin {3}
116         // user code end
117         handleException(ivjExc);
118     }
119 }
120 /**
121  * Return the HelpDialog property value.
122  * @return javax.swing.JPanel
123  */
124 /* WARNING: THIS METHOD WILL BE REGENERATED. */
125 private javax.swing.JPanel getHelpDialog() {
126     if (ivjHelpDialog == null) {
127         try {
128             ivjHelpDialog = new javax.swing.JPanel();
129             ivjHelpDialog.setName("HelpDialog");
130             ivjHelpDialog.setLayout(new java.awt.BorderLayout());
131             getHelpDialog().add(getJPanel1(), "South");
132             getHelpDialog().add(getJSplitPane1(), "Center");
133             // user code begin {1}
134             ivjHelpDialog.setBorder(javax.swing.BorderFactory.createMatteBorder(6,6,6,6,ivjHelpDialog.
135                 getBackground()));
136             // user code end
137         } catch (java.lang.Throwable ivjExc) {
138             // user code begin {2}
139             // user code end
140             handleException(ivjExc);
141         }
142     }
143     return ivjHelpDialog;
144 }
145 /**
146  * Return the JPanel1 property value.
147  * @return javax.swing.JPanel
148  */
149 /* WARNING: THIS METHOD WILL BE REGENERATED. */
150 private javax.swing.JPanel getJPanel1() {
151     if (ivjJPanel1 == null) {
152         try {
153             ivjJPanel1 = new javax.swing.JPanel();
154             ivjJPanel1.setName("JPanel1");
155             ivjJPanel1.setLayout(getJPanel1FlowLayout());
156         }
157     }
158 }
```

```

152         getJPanel1().add(getOKButton(), getOKButton().getName());
153         // user code begin {1}
154         // user code end
155     } catch (java.lang.Throwable ivjExc) {
156         // user code begin {2}
157         // user code end
158         handleException(ivjExc);
159     }
160     return ivj JPanel1;
161 }
162 /**
163 * Return the JPanel1FlowLayout property value.
164 * @return java.awt.FlowLayout
165 */
166 /* WARNING: THIS METHOD WILL BE REGENERATED. */
167 private java.awt.FlowLayout getJPanel1FlowLayout() {
168     java.awt.FlowLayout ivj JPanel1FlowLayout = null;
169     try {
170         /* Create part */
171         ivj JPanel1FlowLayout = new java.awt.FlowLayout();
172         ivj JPanel1FlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
173     } catch (java.lang.Throwable ivjExc) {
174         handleException(ivjExc);
175     };
176     return ivj JPanel1FlowLayout;
177 }
178 /**
179 * Return the JSplitPane1 property value.
180 * @return javax.swing.JSplitPane
181 */
182 /* WARNING: THIS METHOD WILL BE REGENERATED. */
183 private javax.swing.JSplitPane getJSplitPane1() {
184     if (ivj JSplitPane1 == null) {
185         try {
186             ivj JSplitPane1 = new javax.swing.JSplitPane(javax.swing.JSplitPane.VERTICAL_SPLIT);
187             ivj JSplitPane1.setName("JSplitPane1");
188             getJSplitPane1().add(getJTextPane1(), "bottom");
189             getJSplitPane1().add(getTopicTree(), "top");
190             // user code begin {1}
191             // user code end
192         } catch (java.lang.Throwable ivjExc) {
193             // user code begin {2}
194             // user code end
195             handleException(ivjExc);
196         }
197     }
198     return ivj JSplitPane1;
199 }
200 /**
201 * Return the JTextPane1 property value.
202 * @return javax.swing.JTextPane
203 */
204 /* WARNING: THIS METHOD WILL BE REGENERATED. */
205 private javax.swing.JTextPane getJTextPane1() {
206     if (ivj JTextPane1 == null) {
207         try {
208             ivj JTextPane1 = new javax.swing.JTextPane();
209             ivj JTextPane1.setName("JTextPane1");
210             // user code begin {1}
211             // user code end
212         } catch (java.lang.Throwable ivjExc) {
213             // user code begin {2}
214             // user code end
215             handleException(ivjExc);
216         }

```

```

        }
218     return ivjJTextPanel1;
}
220 /**
 * Return the OKButton property value.
221 * @return javax.swing.JButton
 */
224 /* WARNING: THIS METHOD WILL BE REGENERATED. */
225 private javax.swing.JButton getOKButton() {
226     if (ivjOKButton == null) {
227         try {
228             ivjOKButton = new javax.swing.JButton();
229             ivjOKButton.setName("OKButton");
230             ivjOKButton.setText("OK");
231             // user code begin {1}
232             // user code end
233         } catch (java.lang.Throwable ivjExc) {
234             // user code begin {2}
235             // user code end
236             handleException(ivjExc);
237         }
238     }
239     return ivjOKButton;
}
240 /**
241 * Return the JTree1 property value.
242 * @return javax.swing.JTree
*/
244 /* WARNING: THIS METHOD WILL BE REGENERATED. */
245 private javax.swing.JTree getTopicTree() {
246     if (ivjTopicTree == null) {
247         try {
248             ivjTopicTree = new javax.swing.JTree();
249             ivjTopicTree.setName("TopicTree");
250             // user code begin {1}
251             ivjTopicTree.setShowsRootHandles( true );
252             ((DefaultTreeModel) ivjTopicTree.getModel() ).setAsksAllowsChildren( true );
253             // user code end
254         } catch (java.lang.Throwable ivjExc) {
255             // user code begin {2}
256             // user code end
257             handleException(ivjExc);
258         }
259     }
260     return ivjTopicTree;
}
262 /**
263 * Called whenever the part throws an exception.
264 * @param exception java.lang.Throwable
*/
266 private void handleException(java.lang.Throwable exception) {
268     /* Uncomment the following lines to print uncaught exceptions to stdout */
269     // System.out.println("----- UNCAUGHT EXCEPTION -----");
270     // exception.printStackTrace(System.out);
}
272 /**
273 * Initializes connections
274 * @exception java.lang.Exception The exception description.
*/
276 /* WARNING: THIS METHOD WILL BE REGENERATED. */
277 private void initConnections() throws java.lang.Exception {
278     // user code begin {1}
279     // user code end
280     getOKButton().addActionListener(ivjEventHandler);
}
282 }

```

```

  /**
  * Initialize the class.
  */
286 /* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
288     try {
289         // user code begin {1}
290         // user code end
291         setName("HelpDialog");
292         setDefaultCloseOperation(javax.WindowConstants.DISPOSE_ON_CLOSE);
293         setSize(426, 240);
294         setTitle("Help");
295         setContentPane(getHelpDialog());
296         initConnections();
297     } catch (java.lang.Throwable ivjExc) {
298         handleException(ivjExc);
299     }
300     // user code begin {2}
301     setLocationRelativeTo( getOwner() );
302     initTree();
303     // user code end
304 }
305 /**
306  * Insert the method's description here.
307  * Creation date: (02-09-00 21:20:16)
308  */
309 public void initTree()
310 {
311     javax.swing.tree.DefaultMutableTreeNode rootNode = new javax.swing.tree.DefaultMutableTreeNode( "Not yet fully implemented" );
312     ((DefaultTreeModel) getTopicTree().getModel() ).setRoot( rootNode );
313 }
314 /**
315  * main entrypoint - starts the part when it is run as an application
316  * @param args java.lang.String[]
317  */
318 public static void main(java.lang.String[] args) {
319     try {
320         HelpDialog aHelpDialog;
321         aHelpDialog = new HelpDialog();
322         aHelpDialog.setModal(true);
323         aHelpDialog.addWindowListener(new java.awt.event.WindowAdapter() {
324             public void windowClosing(java.awt.event.WindowEvent e) {
325                 System.exit(0);
326             }
327         });
328         aHelpDialog.setVisible(true);
329     } catch (Throwable exception) {
330         System.err.println("Exception occurred in main() of javax.swing.JDialog");
331         exception.printStackTrace(System.out);
332     }
333 }
334 }

```

Listing C.44: Message.java

```

package peerviewmisc;
2
import java.io.Serializable;
4 import java.util.Vector;
6 /**
7  * 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)
9  * @author:
10 */
public class Message implements Serializable

```

```

12  {
13      private java.lang.String contents = null;
14      private java.lang.String author = null;
15      private java.lang.String title = null;
16      private java.lang.String messageID = null;
17      private java.lang.String document = null;
18      private java.util.Date creationDate = null;
19      private java.lang.String creator = null;
20
21      private final static java.lang.String[] FIELD_NAMES = { "Author", "Title", "Contents", "MessageID", "DocumentID", "CreatedOn"};
22  /**
23  * Message constructor comment.
24  */
25  public Message() {
26      super();
27  }
28  /**
29  * Insert the method's description here.
30  * Creation date: (10-07-00 00:09:03)
31  * @return java.lang.String
32  */
33  public java.lang.String getAuthor() {
34      return author;
35  }
36  /**
37  * Insert the method's description here.
38  * Creation date: (10-07-00 00:08:45)
39  * @return java.lang.String
40  */
41  public java.lang.String getContents() {
42      return contents;
43  }
44  /**
45  * Insert the method's description here.
46  * Creation date: (16-07-00 10:40:40)
47  * @return java.util.GregorianCalendar
48  */
49  public java.util.Date getCreationDate() {
50      return creationDate;
51  }
52  /**
53  * Insert the method's description here.
54  * Creation date: (19-07-00 17:29:31)
55  * @return java.lang.String
56  */
57  public java.lang.String getCreator() {
58      return creator;
59  }
60  /**
61  * Insert the method's description here.
62  * Creation date: (19-07-00 19:05:13)
63  * @return java.util.Vector
64  */
65  public java.util.Vector getDataAsVector()
66  {
67      Vector v = new Vector();
68
69      v.addElement( getAuthor() );
70      v.addElement( getTitle() );
71      v.addElement( getContents() );
72      v.addElement( getMessageID() );
73      v.addElement( getDocument() );
74      v.addElement( getCreationDate().toString() );
75
76      return v;

```

```

    }
78 /**
 * Insert the method's description here.
80 * Creation date: (10-07-00 01:42:07)
 * @return java.lang.String
82 */
public java.lang.String getDocument() {
84     return document;
}
86 /**
 * Insert the method's description here.
88 * Creation date: (19-07-00 18:14:33)
 * @return java.lang.String[]
90 */
public final static java.lang.String[] getFieldNames() {
92     return FIELD_NAMES;
}
94 /**
 * Insert the method's description here.
96 * Creation date: (10-07-00 00:11:49)
 * @return java.lang.String
98 */
public java.lang.String getMessageID()
100 { // I initially opted for a combination of the toString() method and the document name as unique
     identifier, but
     // the scheme seemed both overly cautious (what is the likelihood of two identically named authors
     // submitting
102    // identically entitled messages at the exact same millisecond ?) and aesthetically unpleasing when
     // presented
     // to the user.
104    return this.toString() + getDocument();
}
106 /**
 * Insert the method's description here.
108 * Creation date: (10-07-00 00:11:02)
 * @return java.lang.String
110 */
public java.lang.String getTitle() {
112     return title;
}
114 /**
 * Insert the method's description here.
116 * Creation date: (10-07-00 00:09:03)
 * @param newAuthor java.lang.String
118 */
public void setAuthor(java.lang.String newAuthor) {
120     author = newAuthor;
}
122 /**
 * Insert the method's description here.
124 * Creation date: (10-07-00 00:08:45)
 * @param newContents java.lang.String
126 */
public void setContents(java.lang.String newContents) {
128     contents = newContents;
}
130 /**
 * Insert the method's description here.
132 * Creation date: (16-07-00 10:40:40)
 * @param newCreationDate java.util.GregorianCalendar
134 */
public void setCreationDate(java.util.Date newCreationDate) {
136     creationDate = newCreationDate;
}
138 /**
 * Insert the method's description here.

```

```

140 * Creation date: (19-07-00 17:29:31)
141 * @param newCreator java.lang.String
142 */
143 public void setCreator(java.lang.String newCreator) {
144     creator = newCreator;
145 }
146 /**
147 * Insert the method's description here.
148 * Creation date: (10-07-00 01:42:07)
149 * @param newDocument java.lang.String
150 */
151 public void setDocument(java.lang.String newDocument) {
152     document = newDocument;
153 }
154 /**
155 * Insert the method's description here.
156 * Creation date: (10-07-00 00:11:02)
157 * @param newTitle java.lang.String
158 */
159 public void setTitle(java.lang.String newTitle) {
160     title = newTitle;
161 }
162 /**
163 * Insert the method's description here.
164 * Creation date: (11-07-00 12:53:01)
165 * @return java.lang.String
166 */
167 public String toString()
168 {
169     return title + ", „by“ " + author + " „on“ " + creationDate.toString();
170 }
171 }
```

Listing C.45: MessageBox.java

```

package peerviewmisc;
1
import java.awt.event.*;
2
3
4 /**
5 * This class implements a box which is used for displaying brief messages.
6 * Creation date: (25-07-00 13:50:50)
7 * @author:
8 */
9 public class MessageBox extends javax.swing.JDialog
10 {
11     public class CloseButtonHandler implements ActionListener
12     {
13         public void actionPerformed( ActionEvent ae )
14         {
15             dispose();
16         }
17     }
18 }
19
20 private javax.swing.JPanel ivjButtonPanel = null;
21 private java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
22 private javax.swing.JButton ivjCloseButton = null;
23 private javax.swing.JLabel ivjIconLabel = null;
24 private javax.swing.JPanel ivjJDialogContentPane = null;
25 private javax.swing.JPanel ivjJDialogContentPane1 = null;
26 private javax.swing.JPanel ivjMainPanel = null;
27     private javax.swing.JTextPane ivjMessagePanel = null;
28 /**
29 * MessageBox constructor comment.
30 */
31 public MessageBox() {
32     super();
```

```

        initialize();
34    }
35 /**
36  * MessageBox constructor comment.
37  * @param owner java.awt.Dialog
38  */
39 public MessageBox(java.awt.Dialog owner) {
40     super(owner);
41 }
42 /**
43  * MessageBox constructor comment.
44  * @param owner java.awt.Dialog
45  * @param title java.lang.String
46  */
47 public MessageBox(java.awt.Dialog owner, String title) {
48     super(owner, title);
49 }
50 /**
51  * MessageBox constructor comment.
52  * @param owner java.awt.Dialog
53  * @param title java.lang.String
54  * @param modal boolean
55  */
56 public MessageBox(java.awt.Dialog owner, String title, boolean modal) {
57     super(owner, title, modal);
58 }
59 /**
60  * MessageBox constructor comment.
61  * @param owner java.awt.Dialog
62  * @param modal boolean
63  */
64 public MessageBox(java.awt.Dialog owner, boolean modal) {
65     super(owner, modal);
66 }
67 /**
68  * MessageBox constructor comment.
69  * @param owner java.awt.Frame
70  */
71 public MessageBox(java.awt.Frame owner) {
72     super(owner);
73 }
74 /**
75  * MessageBox constructor comment.
76  * @param owner java.awt.Frame
77  * @param title java.lang.String
78  */
79 public MessageBox(java.awt.Frame owner, String title) {
80     super(owner, title);
81 }
82 /**
83  * MessageBox constructor comment.
84  * @param owner java.awt.Frame
85  * @param title java.lang.String
86  * @param modal boolean
87  */
88 public MessageBox(java.awt.Frame owner, String title, boolean modal) {
89     super(owner, title, modal);
90 }
91 /**
92  * MessageBox constructor comment.
93  * @param owner java.awt.Frame
94  * @param modal boolean
95  */
96 public MessageBox(java.awt.Frame owner, boolean modal) {
97     super(owner, modal);
98 }

```

```

  /**
100   * Return the ButtonPanel property value.
101   * @return javax.swing.JPanel
102   */
103  /* WARNING: THIS METHOD WILL BE REGENERATED. */
104  private javax.swing.JPanel getButtonPanel() {
105    if (ivjButtonPanel == null) {
106      try {
107        ivjButtonPanel = new javax.swing.JPanel();
108        ivjButtonPanel.setName("ButtonPanel");
109        ivjButtonPanel.setLayout(getButtonPanelFlowLayout());
110        closeButton.add(getCloseButton(), closeButton.getName());
111        // user code begin {1}
112        // user code end
113      } catch (java.lang.Throwable ivjExc) {
114        // user code begin {2}
115        // user code end
116        handleException(ivjExc);
117      }
118    }
119    return ivjButtonPanel;
120  }
121  /**
122   * Return the ButtonPanelFlowLayout property value.
123   * @return java.awt.FlowLayout
124   */
125  /* WARNING: THIS METHOD WILL BE REGENERATED. */
126  private java.awt.FlowLayout getButtonPanelFlowLayout() {
127    java.awt.FlowLayout ivjButtonPanelFlowLayout = null;
128    try {
129      /* Create part */
130      ivjButtonPanelFlowLayout = new java.awt.FlowLayout();
131      ivjButtonPanelFlowLayout.setAlignment(java.awt.FlowLayout.RIGHT);
132    } catch (java.lang.Throwable ivjExc) {
133      handleException(ivjExc);
134    };
135    return ivjButtonPanelFlowLayout;
136  }
137  /**
138   * Return the CloseButton property value.
139   * @return javax.swing.JButton
140   */
141  /* WARNING: THIS METHOD WILL BE REGENERATED. */
142  private javax.swing.JButton getCloseButton() {
143    if (ivjCloseButton == null) {
144      try {
145        ivjCloseButton = new javax.swing.JButton();
146        ivjCloseButton.setName("CloseButton");
147        ivjCloseButton.setMnemonic('C');
148        ivjCloseButton.setText("Close");
149        // user code begin {1}
150        ivjCloseButton.addActionListener( new CloseButtonHandler() );
151        // user code end
152      } catch (java.lang.Throwable ivjExc) {
153        // user code begin {2}
154        // user code end
155        handleException(ivjExc);
156      }
157    }
158    return ivjCloseButton;
159  }
160  /**
161   * Return the IconLabel property value.
162   * @return javax.swing.JLabel
163   */
164  /* WARNING: THIS METHOD WILL BE REGENERATED. */

```

```

    private javax.swing.JLabel getIconLabel() {
166    if (ivjIconLabel == null) {
167        try {
168            ivjIconLabel = new javax.swing.JLabel();
169            ivjIconLabel.setName("IconLabel");
170            ivjIconLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource("/toolbarButtonGraphics/general/Information24.gif")));
171            ivjIconLabel.setText("");
172            // user code begin {1}
173            // user code end
174        } catch (java.lang.Throwable ivjExc) {
175            // user code begin {2}
176            // user code end
177            handleException(ivjExc);
178        }
179    }
180    return ivjIconLabel;
181 }
182 /**
183 * Return the JDialogContentPane property value.
184 * @return javax.swing.JPanel
185 */
186 /* WARNING: THIS METHOD WILL BE REGENERATED. */
187 private javax.swing.JPanel getJDialogContentPane() {
188    if (ivjJDialogContentPane == null) {
189        try {
190            ivjJDialogContentPane = new javax.swing.JPanel();
191            ivjJDialogContentPane.setName("JDialogContentPane");
192            ivjJDialogContentPane.setLayout(new java.awt.BorderLayout());
193            getJDialogContentPane().add(getJDialogContentPanel1(), "Center");
194            // user code begin {1}
195            // user code end
196        } catch (java.lang.Throwable ivjExc) {
197            // user code begin {2}
198            // user code end
199            handleException(ivjExc);
200        }
201    }
202    return ivjJDialogContentPane;
203 }
204 /**
205 * Return the JDialogContentPanel1 property value.
206 * @return javax.swing.JPanel
207 */
208 /* WARNING: THIS METHOD WILL BE REGENERATED. */
209 private javax.swing.JPanel getJDialogContentPanel1() {
210    if (ivjJDialogContentPanel1 == null) {
211        try {
212            ivjJDialogContentPanel1 = new javax.swing.JPanel();
213            ivjJDialogContentPanel1.setName("JDialogContentPanel1");
214            ivjJDialogContentPanel1.setLayout(new java.awt.BorderLayout());
215            getJDialogContentPanel1().add(getButtonPanel(), "South");
216            getJDialogContentPanel1().add(getMainPanel(), "Center");
217            // user code begin {1}
218            // user code end
219        } catch (java.lang.Throwable ivjExc) {
220            // user code begin {2}
221            // user code end
222            handleException(ivjExc);
223        }
224    }
225    return ivjJDialogContentPanel1;
226 }
227 /**
228 * Return the MainPanel property value.
229 * @return javax.swing.JPanel

```

```

230  /*
231  * WARNING: THIS METHOD WILL BE REGENERATED. */
232 private javax.swing.JPanel getMainPanel() {
233     if (ivjMainPanel == null) {
234         try {
235             ivjMainPanel = new javax.swing.JPanel();
236             ivjMainPanel.setName("MainPanel");
237             ivjMainPanel.setLayout(new java.awt.GridBagLayout());
238
239             java.awt.GridBagConstraints constraintsIconLabel = new java.awt.GridBagConstraints();
240             constraintsIconLabel.gridx = 0; constraintsIconLabel.gridy = 0;
241             constraintsIconLabel.anchor = java.awt.GridBagConstraints.WEST;
242             constraintsIconLabel.insets = new java.awt.Insets(4, 4, 4, 4);
243             getMainPanel().add(getIconLabel(), constraintsIconLabel);
244
245             java.awt.GridBagConstraints constraintsMessagePanel = new java.awt.GridBagConstraints();
246             constraintsMessagePanel.gridx = 1; constraintsMessagePanel.gridy = 0;
247             constraintsMessagePanel.fill = java.awt.GridBagConstraints.HORIZONTAL;
248             constraintsMessagePanel.weightx = 1.0;
249             constraintsMessagePanel.weighty = 1.0;
250             constraintsMessagePanel.insets = new java.awt.Insets(4, 4, 4, 4);
251             getMainPanel().add(getMessagePanel(), constraintsMessagePanel);
252             // user code begin {1}
253             ivjMainPanel.setBorder( SharedConstants.createDialogBorder( getMainPanel().getBackground() ) );
254             // user code end
255         } catch (java.lang.Throwable ivjExc) {
256             // user code begin {2}
257             // user code end
258             handleException(ivjExc);
259         }
260     }
261     return ivjMainPanel;
262 }
263 /**
264  * Return the MessagePanel property value.
265  * @return javax.swing.JTextPane
266  */
267 /* WARNING: THIS METHOD WILL BE REGENERATED. */
268 private javax.swing.JTextPane getMessagePanel() {
269     if (ivjMessagePanel == null) {
270         try {
271             ivjMessagePanel = new javax.swing.JTextPane();
272             ivjMessagePanel.setName("MessagePanel");
273             ivjMessagePanel.setFont(new java.awt.Font("sansserif.bold", 1, 12));
274             ivjMessagePanel.setBackground(java.awt.Color.lightGray);
275             ivjMessagePanel.setEditable(false);
276             // user code begin {1}
277             ivjMessagePanel.setBackground( getButtonPanel().getBackground() );
278             // user code end
279         } catch (java.lang.Throwable ivjExc) {
280             // user code begin {2}
281             // user code end
282             handleException(ivjExc);
283         }
284     }
285     return ivjMessagePanel;
286 }
287 /**
288  * Called whenever the part throws an exception.
289  * @param exception java.lang.Throwable
290  */
291 private void handleException(java.lang.Throwable exception) {
292
293     /* Uncomment the following lines to print uncaught exceptions to stdout */
294     // System.out.println("----- UNCAUGHT EXCEPTION -----");
295     // exception.printStackTrace(System.out);

```

```

296 }
297 /**
298 * Initialize the class.
299 */
300 /* WARNING: THIS METHOD WILL BE REGENERATED. */
301 private void initialize() {
302     try {
303         // user code begin {1}
304         // user code end
305         setName("MessageBox");
306         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
307         setSize(390, 245);
308         setModal(true);
309         setTitle("MessageBox");
310         setContentPane(getJDialogContentPane());
311     } catch (java.lang.Throwable ivjExc) {
312         handleException(ivjExc);
313     }
314     // user code begin {2}
315     // user code end
316 }
317 /**
318 * main entrypoint - starts the part when it is run as an application
319 * @param args java.lang.String[]
320 */
321 public static void main(java.lang.String[] args) {
322     try {
323         MessageBox aMessageBox;
324         aMessageBox = new MessageBox();
325         aMessageBox.setModal(true);
326         aMessageBox.addWindowListener(new java.awt.event.WindowAdapter() {
327             public void windowClosing(java.awt.event.WindowEvent e) {
328                 System.exit(0);
329             };
330         });
331         aMessageBox.setVisible(true);
332     } catch (Throwable exception) {
333         System.err.println("Exception occurred in main() of javax.swing.JDialog");
334         exception.printStackTrace(System.out);
335     }
336 }
337 /**
338 * Insert the method's description here.
339 * Creation date: (25-07-00 14:22:09)
340 * @param newText java.lang.String
341 */
342 public void setText(String newText)
343 {
344     getMessagePanel().setText( newText );
345     getMessagePanel().setPreferredSize( SharedConstants.computeDisplayDimension( newText, getMessagePanel()
346         .getFontMetrics( getMessagePanel().getFont() ) ) );
347     getMessagePanel().setSize( getMessagePanel().getPreferredSize() );
348     getMessagePanel().setAlignmentY( javax.swing.JComponent.CENTER_ALIGNMENT );
349     pack();
350     setLocationRelativeTo( getOwner() );
351 }

```

Listing C.46: NewGroup.java

```

2 package peerviewmisc;
3
4 import java.awt.event.*;
5
6 /**
7 * The dialog box for specifying and submitting a new group.
8 * Creation date: (29-06-00 15:52:46)

```

```

8   * @author:
9   */
10  public class NewGroup extends DisplayGroup
11  {
12
13  /**
14   * NewGroup constructor comment.
15   */
16  public NewGroup() {
17      super();
18      initialize();
19  }
20 /**
21  * NewGroup constructor comment.
22  * @param owner java.awt.Dialog
23  */
24  public NewGroup(java.awt.Dialog owner) {
25      super(owner);
26  }
27 /**
28  * NewGroup constructor comment.
29  * @param owner java.awt.Dialog
30  * @param title java.lang.String
31  */
32  public NewGroup(java.awt.Dialog owner, String title) {
33      super(owner, title);
34  }
35 /**
36  * NewGroup constructor comment.
37  * @param owner java.awt.Dialog
38  * @param title java.lang.String
39  * @param modal boolean
40  */
41  public NewGroup(java.awt.Dialog owner, String title, boolean modal) {
42      super(owner, title, modal);
43  }
44 /**
45  * NewGroup constructor comment.
46  * @param owner java.awt.Dialog
47  * @param modal boolean
48  */
49  public NewGroup(java.awt.Dialog owner, boolean modal) {
50      super(owner, modal);
51  }
52 /**
53  * NewGroup constructor comment.
54  * @param owner java.awt.Frame
55  */
56  public NewGroup(java.awt.Frame owner) {
57      super(owner);
58  }
59 /**
60  * NewGroup constructor comment.
61  * @param owner java.awt.Frame
62  * @param title java.lang.String
63  */
64  public NewGroup(java.awt.Frame owner, String title) {
65      super(owner, title);
66  }
67 /**
68  * NewGroup constructor comment.
69  * @param owner java.awt.Frame
70  * @param title java.lang.String
71  * @param modal boolean
72  */
73  public NewGroup(java.awt.Frame owner, String title, boolean modal) {

```

```

74     super(owner, title, modal);
75 }
76 /**
77 * NewGroup constructor comment.
78 * @param owner java.awt.Frame
79 * @param modal boolean
80 */
81 public NewGroup(java.awt.Frame owner, boolean modal) {
82     super(owner, modal);
83 }
84 /**
85 * Called whenever the part throws an exception.
86 * @param exception java.lang.Throwable
87 */
88 private void handleException(java.lang.Throwable exception) {
89
90     /* Uncomment the following lines to print uncaught exceptions to stdout */
91     // System.out.println("----- UNCAUGHT EXCEPTION -----");
92     // exception.printStackTrace(System.out);
93 }
94 /**
95 * Initialize the class.
96 */
97 /* WARNING: THIS METHOD WILL BE REGENERATED. */
98 private void initialize() {
99     try {
100         // user code begin {1}
101         // user code end
102         setName("NewGroup");
103         setModal(true);
104     } catch (java.lang.Throwable ivjExc) {
105         handleException(ivjExc);
106     }
107     // user code begin {2}
108     setTitle( "NewGroup" );
109     editable( true );
110     getOKButton().setText( "Submit" );
111     getOKButton().setMnemonic( 's' );
112     getCancelButton().setText( "Discard" );
113     getCancelButton().setMnemonic( 'd' );
114
115     // user code end
116 }
117 /**
118 * main entrypoint - starts the part when it is run as an application
119 * @param args java.lang.String[]
120 */
121 public static void main(java.lang.String[] args) {
122     try {
123         NewGroup aNewGroup;
124         aNewGroup = new NewGroup();
125         aNewGroup.setModal(true);
126         aNewGroup.addWindowListener(new java.awt.event.WindowAdapter() {
127             public void windowClosing(java.awt.event.WindowEvent e) {
128                 System.exit(0);
129             };
130         });
131         aNewGroup.setVisible(true);
132     } catch (Throwable exception) {
133         System.err.println("Exception occurred in main() of serverapp.NewGroup");
134         exception.printStackTrace(System.out);
135     }
136 }
137 /**
138 * Insert the method's description here.
139 * Creation date: (04-07-00 00:15:52)

```

```

140  /*
141  public void submitAndClose()
142  {
143      String groupName = ivjGroupNameField.getText();
144      String description = ivjDescriptionField.getText();
145      ErrorBox errorBox = new ErrorBox();
146
147      if ( groupName.length() == 0 )
148      {
149          errorBox.setText( SharedConstants.getGROUP_NAME_MISSING_MESSAGE() );
150          errorBox.show();
151          return;
152      }
153      if ( getGroupDirectory().groupExists( groupName, description )== true )
154      {
155          errorBox.setText( SharedConstants.getGROUP_EXISTS_MESSAGE() );
156          errorBox.show();
157          return;
158      }
159      getWorkGroup().setDescription( ivjDescriptionField.getText() );
160      getWorkGroup().setGroupName( ivjGroupNameField.getText() );
161      setIsDiscarded( false );
162      dispose();
163  }
164 }
```

Listing C.47: Pair.java

```

package peerviewmisc;
2
 /**
4  * A container primitive mapping a key to a value.
5  * Creation date: (20-09-00 02:31:55)
6  * @author:
7  */
8  public class Pair implements java.io.Serializable {
9      private java.lang.String key = null;
10     private java.lang.String value = null;
11
12     * Pair constructor comment.
13     */
14    public Pair() {
15        super();
16    }
17
18    * Insert the method's description here.
19    * Creation date: (20-09-00 02:36:16)
20    * @param keyArg java.lang.String
21    * @param valueArg java.lang.String
22    */
23    public Pair(String keyArg, String valueArg)
24    {
25        key = keyArg;
26        value = valueArg;
27    }
28
29    * Insert the method's description here.
30    * Creation date: (20-09-00 02:32:13)
31    * @return java.lang.String
32    */
33    public java.lang.String getKey() {
34        return key;
35    }
36
37    * Insert the method's description here.
38    * Creation date: (20-09-00 02:34:30)
39    * @return java.lang.String
```

```

40  /*
41   * Insert the method's description here.
42   * Creation date: (20-09-00 02:32:13)
43   * @param newKey java.lang.String
44   */
45   public void setKey(java.lang.String newKey) {
46     key = newKey;
47   }
48 /**
49  * Insert the method's description here.
50  * Creation date: (20-09-00 02:34:30)
51  * @param newValue java.lang.String
52  */
53   public void setValue(java.lang.String newValue) {
54     value = newValue;
55   }
56 }
```

Listing C.48: SharedConstants.java

```

package peerviewmisc;
1
import javax.swing.border.*;
2 import javax.swing.BorderFactory.*;
3 import com.sun.media.jsdt.*;
4 import java.util.Hashtable;
5
6 /**
7  * An entity object containing constants shared by the client and server applications.
8  * Creation date: (30-06-00 21:39:29)
9  * @author:
10 */
11
12 public class SharedConstants {
13   protected final static int BORDER_WIDTH = 6;
14   protected final static java.lang.String GROUP_MANAGEMENT_SESSION_NAME = "group\u00famagement";
15   protected final static java.lang.String DEFAULT_SERVER_NAME = "localhost";
16   protected final static int DEFAULT_SERVER_PORT = 4461;
17   protected final static java.lang.String CONNECTION_TYPE = "Connection\u00e1type";
18   protected final static java.lang.String REQUEST_GROUP_DIRECTORY = "request\u00fagroup\u00fdirectory";
19
20   protected final static java.lang.String SERVER_NAME = "Server\u00fame";
21   protected final static java.lang.String SERVER_PORT = "Server\u00faport";
22   protected final static java.lang.String DEFAULT_CONNECTION_TYPE = "socket";
23   private final static java.lang.String REQUEST_ADD_GROUP = "Request\u00e1add\u00fagroup";
24   private final static java.lang.String GROUP_DIRECTORY = "Group\u00fdirectory";
25   private final static java.lang.String NEW_DOCUMENT = "New\u00fdocument";
26   private final static java.lang.String GROUP_CHANNEL_NAME = "GroupChannel";
27   private final static java.lang.String WORKGROUP_CHANNEL_NAME = "Workgroup\u00fdchannel";
28   private final static java.lang.String CLIENT_ADDED_DOCUMENTS = "Client\u00e1added\u00fddocuments";
29   private final static java.lang.String REQUEST_TOPIC_TREE = "Request\u00e1topic\u00fdtree";
30   private final static java.lang.String TOPIC_TREE = "Topic\u00fdtree";
31   private final static java.lang.String REQUEST_MESSAGE = "Request\u00e1message";
32   private final static java.lang.String MESSAGE = "Message";
33   private final static java.lang.String DISCUSSION_UPDATE_FROM_CLIENT = "Discussion\u00e1update\u00f4from\u00f4client"
34   ;
35   private final static java.lang.String CLIENT_ADDED_MESSAGE = "Client\u00e1added\u00fdmessage";
36   private final static java.lang.String TOPIC_TREE_UPDATE = "Topic\u00fdtree\u00e1update";
37   private final static java.lang.String DELETE_MESSAGE = "Delete\u00fdmessage";
38   private final static java.lang.String CLIENT_REMOVED_LOCAL_DOCUMENTS = "Client\u00e1removed\u00fdlocal\u00fd
39   documents";
40   public final static java.lang.String CLIENT_REMOVED_DOCUMENT = "Client\u00e1removed\u00fddocument";
41   protected final static java.lang.String DEFAULT_CLIENT_NAME = "defaultName";
42   private final static java.lang.String WORKGROUP_UPDATE_FROM_CLIENT = "Workgroup\u00e1update\u00f4from\u00f4client";
```

```

42     private final static java.lang.String WORKGROUP_UPDATE = "Workgroup\u00b7update";
43     private final static java.lang.String CLIENT_LEFT_GROUP = "Client\u00b7left\u00b7group";
44     public final static java.lang.String REMOVE_DOCUMENT = "Document\u00b7was\u00b7removed";
45     public final static java.lang.String GROUP_NAME_MISSING_MESSAGE = "You\u00b7must\u00b7specify\u00b7a\u00b7name\u00b7before\u00b7the
46         \u00b7group\u00b7can\u00b7be\u00b7created.";
47     private final static java.lang.String GROUP_EXISTS_MESSAGE = "A\u00b7group\u00b7by\u00b7this\u00b7name\u00b7and\u00b7description\u00b7
48         already\u00b7exists.";
49     private final static java.lang.String MAXIMUM_PARTICIPANTS_FIELD_INVALID_MESSAGE = "The\u00b7number\u00b7of\u00b7
50         maximum\u00b7participants\u00b7must\u00b7be\u00b7an\u00b7integer\u00b7value\u00b7greater\u00b7than\u00b70.";
51     private final static java.lang.String REQUEST_REMOVE_GROUP = "Remove\u00b7workgroup\u00b7from\u00b7directory";
52
53
54     private final static java.lang.String SOCKET = "socket";
55     private final static java.lang.String HTTP = "http";
56     private final static java.lang.String[] CONNECTION_TYPES = {getSOCKET(),getHTTP()};
57
58     private final static java.lang.String[][] JSDT_EXCEPTION_MESSAGE = {
59         { new AlreadyBoundException().getClass().toString(), "An\u00b7attempt\u00b7to\u00b7register\u00b7a\u00b7group\u00b7or\u00b7group\u00b7
60             member\u00b7conflicted\u00b7with\u00b7one\u00b7or\u00b7more\u00b7existing\u00b7entries" },
61         { new ConnectionException().getClass().toString(), "A\u00b7connection\u00b7error\u00b7occurred\u00b7while\u00b7attempting\u00b7
62             to\u00b7connect\u00b7to\u00b7another\u00b7member\u00b7or\u00b7the\u00b7server" },
63         { new InvalidClientException().getClass().toString(), "An\u00b7attempt\u00b7to\u00b7access\u00b7or\u00b7reference\u00b7a\u00b7group\u00b7
64             member\u00b7resulted\u00b7in\u00b7failure" },
65         { new InvalidURLException().getClass().toString(), "Tried\u00b7to\u00b7use\u00b7a\u00b7malformed\u00b7URL\u00b7for\u00b7network\u00b7
66             access" },
67         { new NameInUseException().getClass().toString(), "There\u00b7was\u00b7a\u00b7naming\u00b7conflict\u00b7between\u00b7two\u00b7or\u00b7more
68             \u00b7Peerview\u00b7entities" },
69         { new NoRegistryException().getClass().toString(), "Could\u00b7not\u00b7connect\u00b7to\u00b7the\u00b7registry" },
70         { new NoSuchChannelException().getClass().toString(), "An\u00b7attempt\u00b7was\u00b7made\u00b7to\u00b7access\u00b7a\u00b7non-
71             existent\u00b7group\u00b7communications\u00b7channel" },
72         { new NoSuchClientException().getClass().toString(), "An\u00b7attempt\u00b7was\u00b7made\u00b7to\u00b7reference\u00b7or\u00b7
73             communicate\u00b7with\u00b7a\u00b7non\u00b7-existent\u00b7group\u00b7member" },
74         { new NoSuchConsumerException().getClass().toString(), "An\u00b7attempt\u00b7was\u00b7made\u00b7to\u00b7reference\u00b7or\u00b7
75             communicate\u00b7with\u00b7a\u00b7non\u00b7-existent\u00b7group" },
76         { new NoSuchHostException().getClass().toString(), "Could\u00b7not\u00b7connect\u00b7to\u00b7the\u00b7specified\u00b7server" },
77         { new NoSuchSessionException().getClass().toString(), "An\u00b7attempt\u00b7was\u00b7made\u00b7to\u00b7reference\u00b7or\u00b7
78             communicate\u00b7with\u00b7a\u00b7non\u00b7-existent\u00b7group" },
79     };
80     private final static java.util.Hashtable JSDT_EXCEPTION_MESSAGES_TABLE = new Hashtable();
81     private final static int NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS = 50;
82     private final static java.lang.String COMPRESSION_LEVEL = "Compression\u00b7level";
83     private final static int DEFAULT_COMPRESSION_LEVEL = java.util.zip.Deflater.BEST_COMPRESSION;
84     private final static java.lang.String DISCUSSIONS_FILE_NAME = "discuss.dat";
85     private final static java.lang.String SOCKET_ARGUMENT = "-socket";
86     private final static java.lang.String HTTP_ARGUMENT = "-http";
87
88     private final static java.lang.String REGISTRY_PORT = "Registry\u00b7port";
89     private final static java.lang.String DEFAULT_REGISTRY_PORT = "4561";
90     private final static java.lang.String WORKGROUP_INCONSISTENT = "Workgroup\u00b7inconsistent";
91     private final static java.lang.String CLIENT_JOINED_GROUP = "Client\u00b7joined\u00b7group";
92     private final static java.lang.String NEW_DOCUMENTS = "New\u00b7documents";
93     private final static java.lang.String REMOVE_DOCUMENTS = "Remove\u00b7documents";
94     private final static java.lang.String CLIENT_LEFT = "Client\u00b7left";
95     private final static java.lang.String COULD_NOT_EXIT_PROPERLY = "The\u00b7application\u00b7could\u00b7not\u00b7be\u00b7
96         properly\u00b7terminated.\u00b7There\u00b7may\u00b7be\u00b7orphaned\u00b7processes\u00b7running\u00b7on\u00b7your\u00b7system.";
97
98     /**
99      * SharedConstants constructor comment.
100     */

```

```

    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 */
117 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 */
135 public static String createFullyQualifiedDocumentName(String senderName, String documentName)
136 {
    return ("Sender:" + senderName + ".Document:" + documentName);
138 }
139 /**
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 }
147 /**
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 }
155 /**
156 * Insert the method's description here.
157 * Creation date: (11-07-00 21:13:58)
158 * @return java.lang.String
159 */
160 public final static java.lang.String getCLIENT_ADDED_MESSAGE() {
161     return CLIENT_ADDED_MESSAGE;
162 }
163 /**
164 * Insert the method's description here.
165 * Creation date: (19-09-00 16:23:30)
166 * @return java.lang.String
167 */
168 public final static java.lang.String getCLIENT_JOINED_GROUP() {
169     return CLIENT_JOINED_GROUP;
170 }
171 /**
172 * Insert the method's description here.
173 * Creation date: (01-11-00 01:20:01)
174 * @return java.lang.String
175 */
176 public final static java.lang.String getCLIENT_LEFT() {
177     return CLIENT_LEFT;
178 }
179 /**
180 * Insert the method's description here.
181 * Creation date: (26-07-00 01:41:43)
182 * @return java.lang.String
183 */
184 public final static java.lang.String getCLIENT_LEFT_GROUP() {
185     return CLIENT_LEFT_GROUP;
186 }
187 /**
188 * Insert the method's description here.
189 * Creation date: (15-07-00 16:26:04)
190 * @return java.lang.String
191 */
192 public final static java.lang.String getCLIENT_REMOVED_DOCUMENT() {
193     return CLIENT_REMOVED_DOCUMENT;
194 }
195 /**
196 * Insert the method's description here.
197 * Creation date: (15-07-00 15:33:51)
198 * @return java.lang.String
199 */
200 public final static java.lang.String getCLIENT_REMOVED_LOCAL_DOCUMENTS() {
201     return CLIENT_REMOVED_LOCAL_DOCUMENTS;
202 }
203 /**
204 * Insert the method's description here.
205 * Creation date: (19-08-00 12:26:31)
206 * @return java.lang.String
207 */
208 public final static java.lang.String getCOMPRESSION_LEVEL() {
209     return COMPRESSION_LEVEL;
210 }
211 /**
212 * Insert the method's description here.
213 * Creation date: (03-07-00 09:29:42)
214 * @return java.lang.String
215 */
216 public final static java.lang.String getCONNECTION_TYPE() {
217     return CONNECTION_TYPE;
218 }

```

```

  /**
  * Insert the method's description here.
  * Creation date: (05-08-00 20:51:35)
  */
222 public final static java.lang.String[] getConnection_TYPES() {
    return CONNECTION_TYPES;
}
226 /**
  * Insert the method's description here.
  * Creation date: (11/23/00 1:56:30 PM)
  */
230 public final static java.lang.String getCOULD_NOT_EXIT_PROPERLY() {
    return COULD_NOT_EXIT_PROPERLY;
}
234 /**
  * Insert the method's description here.
  * Creation date: (16-07-00 11:32:37)
  */
238 public final static java.lang.String getDefault_CLIENT_NAME() {
    return DEFAULT_CLIENT_NAME;
}
242 /**
  * Insert the method's description here.
  * Creation date: (19-08-00 12:26:58)
  */
246 public final static int getDefault_COMPRESSION_LEVEL() {
    return DEFAULT_COMPRESSION_LEVEL;
}
250 /**
  * Insert the method's description here.
  * Creation date: (03-07-00 16:51:41)
  */
254 public final static java.lang.String getDefault_CONNECTION_TYPE() {
    return DEFAULT_CONNECTION_TYPE;
}
258 /**
  * Insert the method's description here.
  * Creation date: (12-09-00 10:17:15)
  */
262 public final static java.lang.String getDefault_REGISTRY_PORT() {
    return DEFAULT_REGISTRY_PORT;
}
266 /**
  * Insert the method's description here.
  * Creation date: (03-07-00 09:26:36)
  */
270 public final static java.lang.String getDefault_SERVER_NAME() {
    return DEFAULT_SERVER_NAME;
}
274 /**
  * Insert the method's description here.
  * Creation date: (03-07-00 09:28:44)
  */
278 public final static int getDefault_SERVER_PORT() {
    return DEFAULT_SERVER_PORT;
}
282 /**
  * 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.
357 * Creation date: (10-09-00 20:03:32)
358 * @return java.lang.String
359 */
360 public final static java.lang.String getHTTP_ARGUMENT() {
        return HTTP_ARGUMENT;
362 }
/***
364 * Insert the method's description here.
365 * Creation date: (08-08-00 16:57:01)
366 * @return java.lang.String[]
367 */
368 public final static java.lang.String[][] getJSDT_EXCEPTION_MESSAGE() {
        return JSDT_EXCEPTION_MESSAGE;
370 }
/***
372 * Insert the method's description here.
373 * Creation date: (09-08-00 07:59:38)
374 * @return java.util.Hashtable
375 */
376 public final static java.util.Hashtable getJSDT_EXCEPTION_MESSAGES_TABLE()
{
378     if ( JSDT_EXCEPTION_MESSAGES_TABLE.size() == 0 )
379     {
380         for ( int i = 0 ; i < JSDT_EXCEPTION_MESSAGE.length; i++ )
381         {
382             JSDT_EXCEPTION_MESSAGES_TABLE.put( JSDT_EXCEPTION_MESSAGE[i][0] , JSDT_EXCEPTION_MESSAGE[i][1] );
383         }
384     }
385     return JSDT_EXCEPTION_MESSAGES_TABLE;
386 }
/***
388 * Insert the method's description here.
389 * Creation date: (26-07-00 16:57:40)
390 * @return java.lang.String
391 */
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.
397 * Creation date: (10-07-00 02:26:24)
398 * @return java.lang.String
399 */
400 public final static java.lang.String getMessage() {
        return MESSAGE;
402 }
/***
404 * Insert the method's description here.
405 * Creation date: (06-07-00 23:45:33)
406 * @return java.lang.String
407 */
408 public final static java.lang.String getNEW_DOCUMENT() {
        return NEW_DOCUMENT;
410 }
/***
412 * Insert the method's description here.
413 * Creation date: (19-09-00 18:28:18)
414 * @return java.lang.String
415 */
416 public final static java.lang.String getNEW_DOCUMENTS() {

```

```

        return NEW_DOCUMENTS;
418 }
419 /**
420 * Insert the method's description here.
421 * Creation date: (10-08-00 06:56:22)
422 * @return int
423 */
424 public final static int getNOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS() {
425     return NOTIFICATION_MESSAGE_PREFERRED_WIDTH_IN_CHARACTERS;
426 }
427 /**
428 * Insert the method's description here.
429 * Creation date: (12-09-00 10:16:20)
430 * @return java.lang.String
431 */
432 public final static java.lang.String getREGISTRY_PORT() {
433     return REGISTRY_PORT;
434 }
435 /**
436 * Insert the method's description here.
437 * Creation date: (26-07-00 02:41:39)
438 * @return java.lang.String
439 */
440 public final static java.lang.String getREMOVE_DOCUMENT() {
441     return REMOVE_DOCUMENT;
442 }
443 /**
444 * Insert the method's description here.
445 * Creation date: (19-09-00 19:15:50)
446 * @return java.lang.String
447 */
448 public final static java.lang.String getREMOVE_DOCUMENTS() {
449     return REMOVE_DOCUMENTS;
450 }
451 /**
452 * Insert the method's description here.
453 * Creation date: (04-07-00 17:33:01)
454 * @return java.lang.String
455 */
456 public final static java.lang.String getRequest_ADD_GROUP() {
457     return REQUEST_ADD_GROUP;
458 }
459 /**
460 * Insert the method's description here.
461 * Creation date: (03-07-00 12:39:46)
462 * @return java.lang.String
463 */
464 public final static java.lang.String getRequest_GROUP_DIRECTORY() {
465     return REQUEST_GROUP_DIRECTORY;
466 }
467 /**
468 * Insert the method's description here.
469 * Creation date: (10-07-00 02:16:46)
470 * @return java.lang.String
471 */
472 public final static java.lang.String getRequest_MESSAGE() {
473     return REQUEST_MESSAGE;
474 }
475 /**
476 * Insert the method's description here.
477 * Creation date: (27-07-00 00:26:45)
478 * @return java.lang.String
479 */
480 public final static java.lang.String getRequest_REMOVE_GROUP() {
481     return REQUEST_REMOVE_GROUP;
482 }

```

```

    /**
484 * Insert the method's description here.
485 * Creation date: (09-07-00 21:54:55)
486 * @return java.lang.String
487 */
488 public final static java.lang.String getREQUEST_TOPIC_TREE() {
489     return REQUEST_TOPIC_TREE;
490 }
491 /**
492 * Insert the method's description here.
493 * Creation date: (03-07-00 14:30:27)
494 * @return java.lang.String
495 */
496 public final static java.lang.String getSERVER_NAME() {
497     return SERVER_NAME;
498 }
499 /**
500 * Insert the method's description here.
501 * Creation date: (03-07-00 14:30:42)
502 * @return java.lang.String
503 */
504 public final static java.lang.String getSERVER_PORT() {
505     return SERVER_PORT;
506 }
507 /**
508 * Insert the method's description here.
509 * Creation date: (10-09-00 20:03:50)
510 * @return java.lang.String
511 */
512 public final static java.lang.String getSOCKET() {
513     return SOCKET;
514 }
515 /**
516 * Insert the method's description here.
517 * Creation date: (10-09-00 20:03:13)
518 * @return java.lang.String
519 */
520 public final static java.lang.String getSOCKET_ARGUMENT() {
521     return SOCKET_ARGUMENT;
522 }
523 /**
524 * Insert the method's description here.
525 * Creation date: (09-07-00 22:29:03)
526 * @return java.lang.String
527 */
528 public final static java.lang.String getTOPIC_TREE() {
529     return TOPIC_TREE;
530 }
531 /**
532 * Insert the method's description here.
533 * Creation date: (11-07-00 21:31:23)
534 * @return java.lang.String
535 */
536 public final static java.lang.String getTOPIC_TREE_UPDATE() {
537     return TOPIC_TREE_UPDATE;
538 }
539 /**
540 * Insert the method's description here.
541 * Creation date: (08-07-00 17:28:19)
542 * @return java.lang.String
543 */
544 public final static java.lang.String getWORKGROUP_CHANNEL_NAME() {
545     return WORKGROUP_CHANNEL_NAME;
546 }
547 /**
548 * Insert the method's description here.

```

```

550 * Creation date: (18-09-00 20:53:55)
551 * @return java.lang.String
552 */
552 public final static java.lang.String getWORKGROUP_INCONSISTENT() {
553     return WORKGROUP_INCONSISTENT;
554 }
555 /**
556 * Insert the method's description here.
557 * Creation date: (22-07-00 20:15:39)
558 * @return java.lang.String
559 */
560 public final static java.lang.String getWORKGROUP_UPDATE() {
561     return WORKGROUP_UPDATE;
562 }
563 /**
564 * Insert the method's description here.
565 * Creation date: (22-07-00 17:52:54)
566 * @return java.lang.String
567 */
568 public final static java.lang.String getWORKGROUP_UPDATE_FROM_CLIENT() {
569     return WORKGROUP_UPDATE_FROM_CLIENT;
570 }
571 }
```

Listing C.49: WorkGroup.java

```

package peerviewmisc;
2
3     import java.io.*;
4     import java.util.Hashtable;
5     import com.sun.media.jsdt.*;
6     import java.util.Vector;
7     import java.util.Date;
8
9 /**
10  * This class implements a representation of a PeerView group.
11  * Creation date: (01-07-00 09:22:26)
12  * @author:
13  */
14 public class WorkGroup implements Serializable
{
15     private java.lang.String groupName = "";
16     private java.lang.String description = "";
17     private com.sun.media.jsdt.URLString sessionURL = null;
18     private long totalDataVolume = 0;
19     private java.util.Date creationDate = new Date();
20     private java.lang.String creator = "";
21     private final static java.lang.String fieldNames[] = { "Group\u005fname", "Description",
22                                         "#\u005fu005fdocuments", "Total\u005fdata\u005fvolume", "Active\u005fparticipants",
23                                         "Creation\u005fdate", "Created\u005fby" };
24
25     private java.util.Hashtable documents = new Hashtable();
26     private java.util.HashSet participants = new java.util.HashSet();
27     private int numberofDocuments = 0;
28 /**
29  * WorkGroup constructor comment.
30  */
31     public WorkGroup() {
32         super();
33     }
34 /**
35  * Insert the method's description here.
36  * Creation date: (19-09-00 19:35:37)
37  * @param workGroup peerviewmisc.WorkGroup
38  */
39     public WorkGroup(WorkGroup workGroup)
40 {
41         setCreationDate( workGroup.getCreationDate() );
42 }
```

```

42     setCreator( workGroup.getCreator() );
43     setDescription( workGroup.getDescription() );
44     setDocuments( (Hashtable) workGroup.getDocuments().clone() );
45     setGroupName( workGroup.getGroupName() );
46     setParticipants( (java.util.HashSet) workGroup.getParticipants().clone() );
47     setSessionURL( workGroup.getSessionURL() );
48     setTotalDataVolume( workGroup.getTotalDataVolume() );
49 }
50 /**
51 * Insert the method's description here.
52 * Creation date: (16-07-00 11:01:38)
53 * @return java.util.Date
54 */
55 public java.util.Date getCreationDate() {
56     return creationDate;
57 }
58 /**
59 * Insert the method's description here.
60 * Creation date: (17-07-00 17:12:13)
61 * @return java.lang.String
62 */
63 public java.lang.String getCreator() {
64     return creator;
65 }
66 /**
67 * Insert the method's description here.
68 * Creation date: (05-07-00 21:32:10)
69 * @return java.lang.String
70 */
71 public java.lang.String getDescription() {
72     return description;
73 }
74 /**
75 * Insert the method's description here.
76 * Creation date: (19-09-00 17:40:00)
77 * @return java.util.Hashtable
78 */
79 public java.util.Hashtable getDocuments() {
80     return documents;
81 }
82 /**
83 * Insert the method's description here.
84 * Creation date: (17-07-00 19:32:07)
85 * @return java.lang.String
86 */
87 public final static java.lang.String[] getFieldNames() {
88     return fieldNames;
89 }
90 /**
91 * Pack the data fields corresponding to the entries in the field label array into a vector and return
92 * it. This method
93 * primarily intended for use with tabular display (a vector corresponds to a row of table data).
94 * Creation date: (17-07-00 20:24:14)
95 * @return java.util.Vector
96 */
97 public java.util.Vector getGroupAsVector()
98 {
99     Vector v = new Vector();
100    v.addElement( getGroupName() );
101    v.addElement( getDescription() );
102    v.addElement( String.valueOf( getNumberOfDocuments() ) );
103    v.addElement( String.valueOf( getTotalDataVolume() ) );
104    v.addElement( String.valueOf( participants.size() ) );
105    v.addElement( getCreationDate().toString() );
106    v.addElement( getCreator() );

```

```

108     return v;
110
112 }
/***
114 * Insert the method's description here.
115 * Creation date: (05-07-00 21:25:46)
116 * @return java.lang.String
117 */
118 public java.lang.String getGroupName() {
119     return groupName;
120 }
/***
122 * Insert the method's description here.
123 * Creation date: (19-09-00 18:45:20)
124 * @return int
125 */
126 public int getNumberOfDocuments() {
127     return numberOfDocuments;
128 }
/***
130 * Insert the method's description here.
131 * Creation date: (19-09-00 17:53:36)
132 * @return java.util.HashSet
133 */
134 public java.util.HashSet getParticipants() {
135     return participants;
136 }
/***
138 * Insert the method's description here.
139 * Creation date: (07-07-00 20:11:38)
140 * @return com.sun.media.jsdt.URLString
141 */
142 public com.sun.media.jsdt.URLString getSessionURL() {
143     return sessionURL;
144 }
/***
146 * Insert the method's description here.
147 * Creation date: (08-07-00 20:06:01)
148 * @return long
149 */
150 public long getTotalDataVolume() {
151     return totalDataVolume;
152 }
/***
154 * Create and return unique identifier for the JSDT channel created to convey information pertaining to
155 * workgroup.
156 * Creation date: (13-08-00 01:42:08)
157 * @return java.lang.String
158 */
159 public String getWorkGroupChannelID()
{
160     return getWorkGroupID() + SharedConstants.getWORKGROUP_CHANNEL_NAME();
}
162 /**
163 * Same as getWorkGroupChannelID(), only with a specifiable suffix.
164 * Creation date: (13-08-00 01:43:08)
165 * @return java.lang.String
166 * @param suffix java.lang.String
167 */
168 public String getWorkGroupChannelID(String suffix)
{
169     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 */
177 public String getWorkGroupID()
178 {
179     return getGroupName() + getDescription() + getCreator();
180 }
181 /**
182 * Insert the method's description here.
 * Creation date: (16-07-00 11:01:38)
184 * @param newCreationDate java.util.Date
185 */
186 public void setCreationDate(java.util.Date newCreationDate) {
187     creationDate = newCreationDate;
188 }
189 /**
190 * Insert the method's description here.
 * Creation date: (17-07-00 17:12:13)
192 * @param newCreator java.lang.String
193 */
194 public void setCreator(java.lang.String newCreator) {
195     creator = newCreator;
196 }
197 /**
198 * Insert the method's description here.
 * Creation date: (05-07-00 21:32:10)
200 * @param newDescription java.lang.String
201 */
202 public void setDescription(java.lang.String newDescription) {
203     description = newDescription;
204 }
205 /**
206 * Insert the method's description here.
 * Creation date: (19-09-00 17:40:00)
208 * @param newDocuments java.util.Hashtable
209 */
210 public void setDocuments(java.util.Hashtable newDocuments) {
211     documents = newDocuments;
212 }
213 /**
214 * Insert the method's description here.
 * Creation date: (05-07-00 21:25:46)
216 * @param newGroupName java.lang.String
217 */
218 public void setGroupName(java.lang.String newGroupName) {
219     groupName = newGroupName;
220 }
221 /**
222 * Insert the method's description here.
 * Creation date: (19-09-00 18:45:20)
224 * @param newNumberOfDocuments int
225 */
226 public void setNumberOfDocuments(int newNumberOfDocuments) {
227     numberOfWorkGroup = newNumberOfDocuments;
228 }
229 /**
230 * Insert the method's description here.
 * Creation date: (19-09-00 17:53:36)
232 * @param newParticipants java.util.HashSet
233 */
234 public void setParticipants(java.util.HashSet newParticipants) {
235     participants = newParticipants;
236 }
237 /**

```

```
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 transcript 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

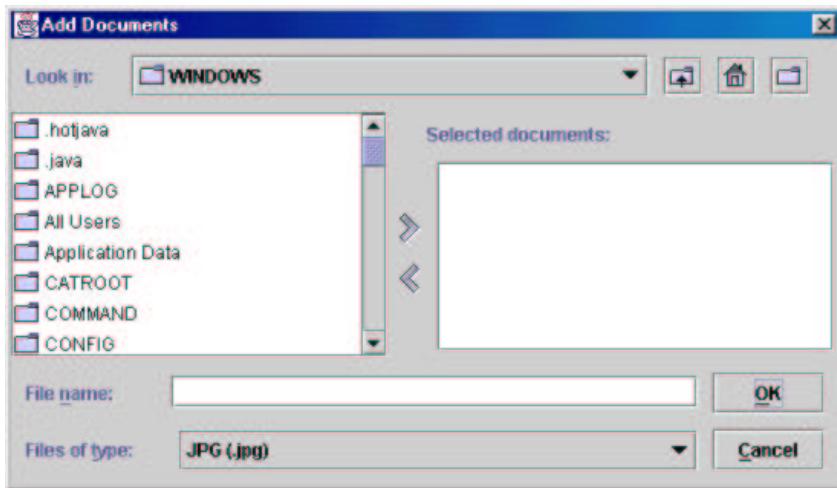
Welcome to the PeerView online help system.

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.

OK

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

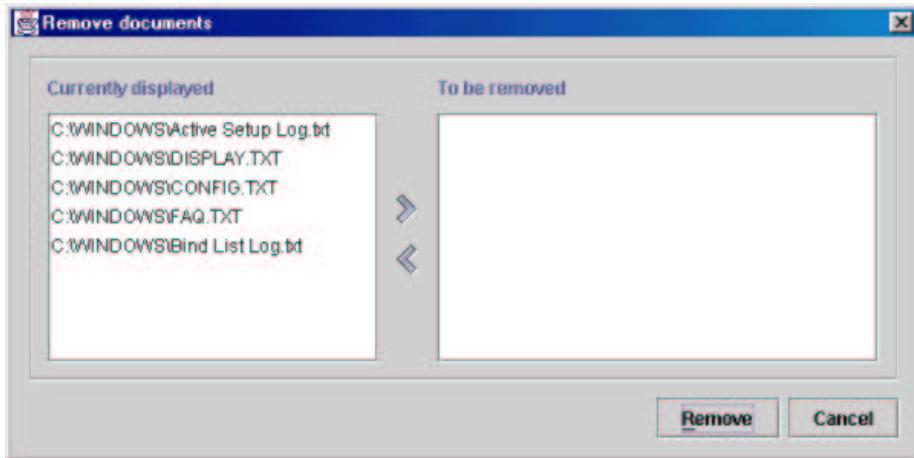
Cancel

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



Close the remove box and remove all documents listed in the "To be removed" box from the panorama.



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.

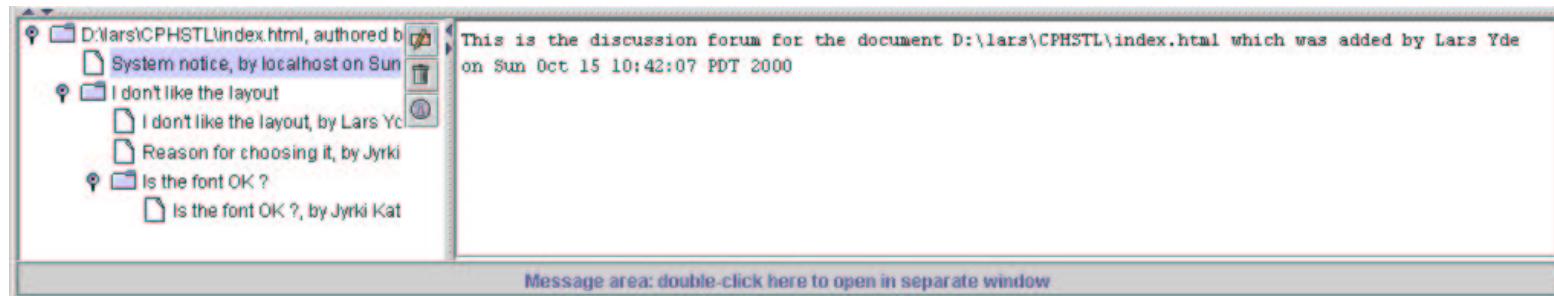
Auto-lock all

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 ♀ icon 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 a small square icon), 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

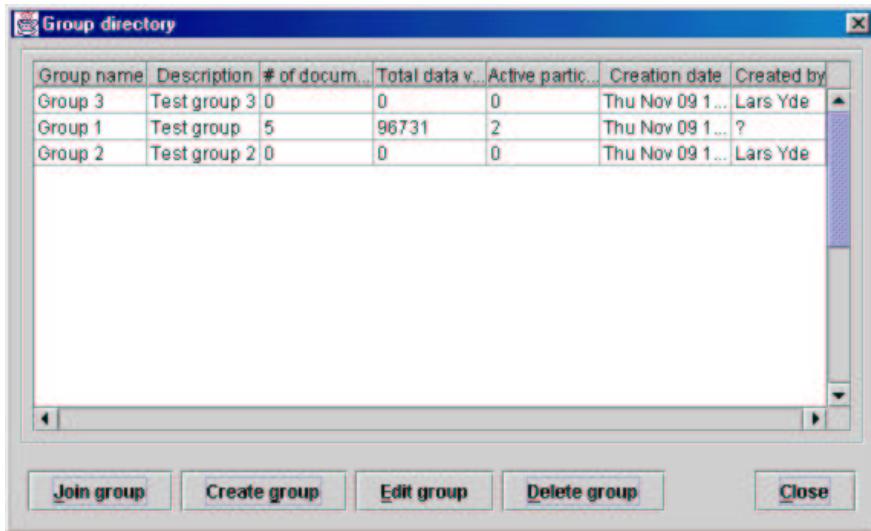
To adjust the size of the discussion forum itself, use the divider bar which separates it from the panorama.

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

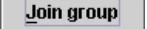
The name of the person who created the group.

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 is 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 group**

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 group**

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.

 **Edit group**

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.

 **Delete group**

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**

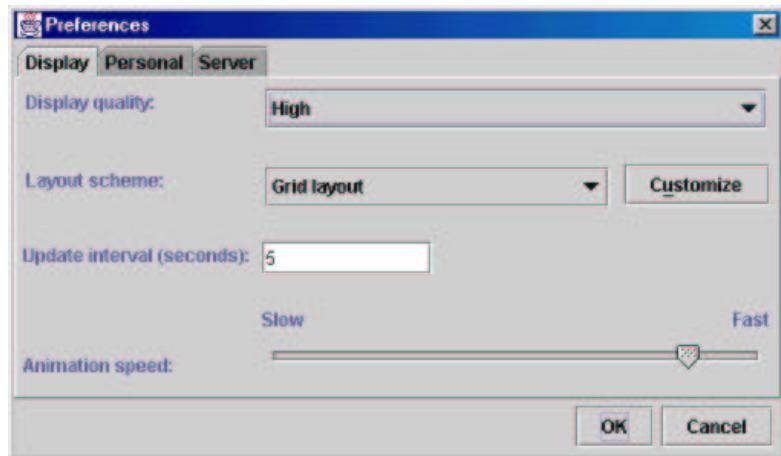
Close the group directory box.

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.

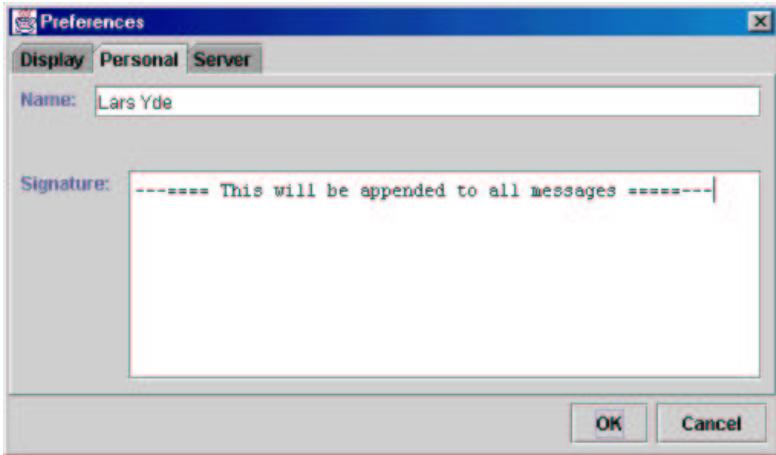
Update interval

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:

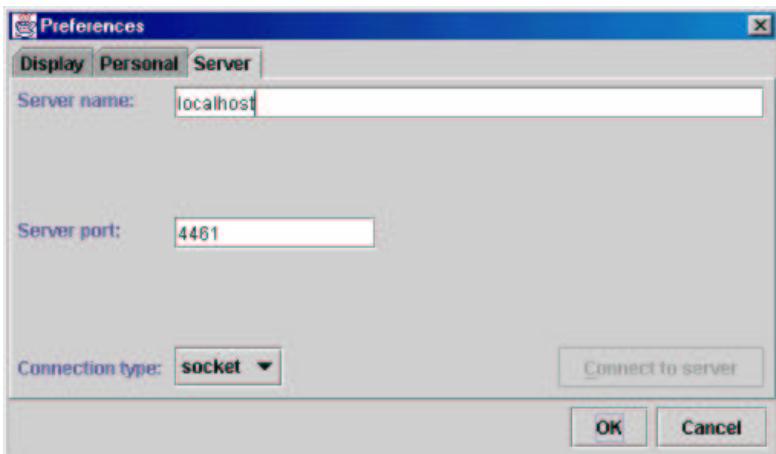
Name

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 remote 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:



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.