Design Report: Resource Management Software

CS400 – Senior Design I

Mark Briggs
Paul Knell
John Schluechtermann
Jeremy Vechinski

Electrical Engineering and Computer Science Department
Milwaukee School of Engineering
Milwaukee, WI, USA

Submitted To: Professor William Barnekow
Submitted: 7 February 2000

Table of Contents

Table of Figures

Introduction

The primary goal of this project is to develop resource allocation management software to ease scheduling of personnel, equipment, and office space across various projects for multiple clients. The proposal outlined the goals of the project, presented a preliminary schedule, and brought up areas in which further research was needed. Critical areas of technology were researched and the findings were summarized in the technology research report. After deciding on specific implementation technologies based on the research, work on the design of the system was begun.

In order to properly design this software system, the procedures outlined in the textbook Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design by Craig Larman were employed. The first step taken was to identify all the tasks a user might ask the system to perform. Use cases (narratives which describe the interaction between the system and the user) were written. Second, by looking at the nouns in the use cases, it was possible to pick out a number of classes for possible implementation. Also, by going back to the proposal and examining other information provided by the client, additional classes and the data to be stored in these classes were obtained. This information was combined to form a conceptual model, which shows the major classes and their attributes.

After the conceptual model was complete, high-level sequence diagrams, which visually depict how the classes interact, were constructed. A sequence diagram was created for each use case. The sequence diagrams aided design in two ways. First, they brought to light several deficiencies in the design. Second, they gave the team an idea of the methods (member functions) that would need to be present in each class. The last step in the design process was to build a complete class diagram. The class diagram lists all the classes that will be implemented in the software system, along with their attributes and methods. It also shows the relationships that exist between the classes.

In parallel with the development above, a specifications document was drafted. The specifications document was constructed to provide a textual description of the features of the system and a general overview of the system implementation. The information covered in the specifications document includes the data to be stored in the database, the graphical user interface elements, and the general operation of the client and server software. The specifications document was read and approved by our client, and will act as a contract regarding the operation of the software. Additionally, design work on the graphical user interface was completed. Layout of all major windows and dialog boxes was completed. Finally, a glossary of terms was built as the design proceeded, ensuring that members of the design team (as well as our client) shared an agreed upon vocabulary.

Use Cases

Use cases are narrative descriptions of domain processes. They describe the sequence of events an actor (user) performs to complete a task. They tell a story about the interaction between a user and the software system. Uses cases help advance the design by aiding in identification of classes, methods, and attributes (Larman 49).

The following use cases were written describing typical functions to be performed by the Resource Management Software. These use cases were written using the requirements of the system as outlined in the specifications document and as presented by our client. Note they are essential use cases, meaning that they describe the user/system interaction without mentioning any specific technologies.

Select Item

Purpose

The purpose of this use case is to allow the user to select an item such as: Client, Project, Resource Allocation, Timeslot, Consultant, Machine, or Room. This use case will be needed as a predecessor to most of the "Modify" and "Delete" use cases.

Overview

Any user will be able to perform this use case. The use case begins when the user selects an item from the list of items that the GUI displays (which will most likely be in the form of a graphical tree component). After selecting an item, the system will verify that the item still exists since it is possible that the user's display is obsolete (out-of-sync with the actual database). If this verification fails, the user's display will be updated. Otherwise, the system will collect the information about the selected item, and display it in view-only mode using the lower-right panel of the GUI. The system will determine whether or not the user has permission to modify the information. If so, a button labeled "Modify" will be displayed. The system will also display the Project-View graph if a project was selected. This would be done by initiating the "Generate Project-View Graph" use case.

Typical Course of Events

Actor Action	System Response
1. This use case begins when a user selects an item shown in the display.	2. The system verified that the item still exists.
	3. The system retrieves information about the selected item. Some information deemed "confidential" will be unavailable based on the type of user.
	4. The system will determine whether or not the user has permission to modify the selected item.
	5. The system will display the information in read-only mode. The "Modify" button will optionally be displayed.

	6. If a project was selected, the "Generate Project-View Graph" use case will be initiated by the system. Otherwise, this use case ends.

Alternative Courses

- Line 2: The item no longer exists. The display is refreshed and the information panel located at the lower-right of the display will be cleared. This use case ends.
- Line 5: The system is unable to access the database. The system displays an error message and the use case ends.

Add Client

Purpose

The purpose of this use case is to add a new client to the system.

Overview

The admin-level user needs to add another client to the system because the client has not done business with the consulting company before, and needs to open a new account in order to begin utilizing the consulting company's services. The information that needs to be entered about the new client consists of: the name of the client's company, the name of the company's managing principal, the name of the company's business development executive, and the status of whether or not the client has signed the Consulting Services Agreement (CSA). The admin-level user will choose the menu item for adding a client. The system will verify that the user actually has permission to perform this action. The system will display dialog box that allows the user to enter the needed information. The default value for the question "Is the CSA signed?" will be false. The user will be able to enter the information and press "Submit", or the user will be able to press "Cancel" to abort the operation. After the user presses "Submit", the system attempt to add the new client to the database and will display a message to indicate success or failure. If successful, the display of the list of clients will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses to add a client.	2. The system checks that the user is allowed to perform this operation.
	3. The system displays a dialog box that allows the user to enter the information about the client.
4. User provides the information and presses "Submit".	5. Program checks to make sure the client name entered does not already exist in the system.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays

	a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add a client. An error message is displayed and the use case ends.
- Line 4: User presses “Cancel” instead of “Submit”. The dialog box closes and the use case ends.
- Line 5: The name already exists for another client in the system. An error message is displayed but the dialog box remains open. Return to line 4 of the typical course of events.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Modify Client

Purpose

The purpose of this use case is to modify a client that already exists in the system.

Overview

The admin-level user needs to modify some of the information that the system keeps track of about a particular client. The information that may be modified consists of any information originally entered in the “Add Client” use case: the client company may undergo a name change, the contact personnel may change, or the status of whether or not the CSA is signed may change. This use case may only begin after the “Select Client” use case has been performed. The user will press the “Modify” button to indicate the user’s desire to modify the selected client. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying the selected client. If it appears that another user is in the process of modifying the selected client, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will display a form (that resembles the “Add Client” form), which allows the information to be changed. The user will be able to press “Submit” to accept the changes, or “Cancel” to abort the operation. If “Submit” was pressed, the system will check that the name was not changed to a name that is already used by another client. The system will update the database. A message will be displayed to indicate success or failure. If the name was changed, the display of the list of clients will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Client” use case, and also presses the “Modify” button.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is currently in the process of modifying (or deleting) the same client.

4. The system will display a form that allows the information to be changed.

5. The user makes the necessary changes and presses “Submit”.

6. The system checks that the name of the client was not changed to the name of another client in the system.

7. The data is updated in the database.

8. The system displays a success/fail message to the user.

9. The display of the list of clients is refreshed if the name changed. This use case ends.

Alternative Courses

Line 2: User does not have permission to modify a client. An error message is displayed and this use case ends.
Line 3: The system finds that another user is currently in the process of modifying the same client. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
Line 5: The user presses “Cancel” instead of “Submit”. In this case, the form, which allows the information to be changed, will close, and this use case ends.
Line 6: The name already exists for another client in the system. An error message is displayed but the dialog box remains open. Return to line 5 of the typical course of events.
Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Delete Client

Purpose

The purpose of this use case is to remove a client from the system.

Overview

The admin-level user may need to remove a client from the system. Clients will only be removable if they have no associated projects. Usually, this use case will be performed if a client was added accidentally. It could also be used to purge information from the database (if it gets too large), but this would most likely not be done since it is desired to keep a history of all current and terminated projects. This use case may only begin after the “Select Client” use case has been performed. The user will select “Delete Client” from the menu. The system will verify that the user actually has permission to perform this action. The system will check that the selected client is not associated with any projects. This is done to avoid the situation of having a project that refers to a client that no longer exists. The system will verify that no other user is in the process of modifying the selected client. If it appears that another user is in the process of modifying the selected client, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will prompt the user for confirmation. The system will remove the client from the system, display a success/fail message box, and then refresh the display of the list of clients (if the operation was successful).

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Client” use case, and also selects “Delete Client” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that this client has no associated projects.
	4. The system checks that no other user is in the process of modifying (or deleting) the same client.
	5. System prompts for confirmation.
6. User presses the “Delete” button of the confirmation dialog box.	7. The system removes the client from the database.
	8. The system displays a success/fail message box.
	9. The system refreshes the display.

Alternative Courses

Line 2: User does not have permission to delete a client. An error message is displayed and this use case ends.
Line 3: The selected client has projects associated with it. An error message is displayed and this use case ends.
Line 4: The system finds that another user is currently in the process of modifying the same client. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 5 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
Line 6: The user presses “Cancel” instead of “Delete”. The use case ends.
Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Add Resource

Purpose

The purpose of this use case is to add a new resource to the system.

Overview

This use case varies slightly based on the type of resource the user needs to add. The possible types of resources that the user may add are: Consultant, Machine, and Room. Only an admin-level user may perform this use case. The admin-level user needs to add another resource to the system in the event that another consultant was hired, another machine was purchased, or an office was expanded providing additional rooms. The information that needs to be entered about a consultant consists of: the consultant's first, middle, and last name; e-mail address; office where the consultant works; cost per hour; cost per overtime hour; target rate and target overtime rate. The information that needs to be entered about a machine consists of: the machine's serial number, name, description, cost per hour, cost per overtime hour, target rate and target overtime rate. The information that needs to be entered about a room consists of: the room's name, description, office, target rate, target overtime rate, cost, and overtime cost. The cost fields will be used to determine default values (based on a known formula) for the target rate fields. The admin-level user will choose the menu item for adding a particular type of resource (consultant, machine, or room). The system will verify that the user actually has permission to perform this action. The system will display a dialog box that allows the user to enter the needed information. The user will be able to enter the information and press "Submit", or the user will be able to press "Cancel" to abort the operation. After the user presses "Submit", the system attempt to add the new resource to the database and will display a message to indicate success or failure. If successful, the display of the list of resources will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses to add a particular type of resource.	2. The system checks that the user is allowed to perform this operation.
	3. The system displays a dialog box that allows the user to enter the information about the resource.
4. User provides the information and presses "Submit".	5. Program checks to make sure that there isn't already another resource of the same type with the same name.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add this particular type of resource. An error message is displayed and the use case ends.
- Line 4: User presses "Cancel" instead of "Submit". The dialog box closes and the use case ends.

- Line 5: The name already exists for another client in the system. An error message is displayed but the dialog box remains open. Return to line 4 of the typical course of events.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Modify Resource

Purpose

The purpose of this use case is to modify a resource that already exists in the system.

Overview

The admin-level user needs to modify some of the information that the system keeps track of about a particular resource. The information that may be modified consists of any information originally entered in the “Add Resource” use case. This use case may only begin after the “Select Item” use case has been performed using a resource as the item being selected. The user will press the “Modify” button to indicate the user’s desire to modify the selected resource. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying the selected resource. If it appears that another user is in the process of modifying the selected resource, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will display a form (that resembles the “Add Resource” form), which allows the information to be changed. The user will be able to press “Submit” to accept the changes, or “Cancel” to abort the operation. If “Submit” was pressed, the system will check that the name was not changed to a name that is already used by another resource of the same type. The system will update the database. A message will be displayed to indicate success or failure. If the name was changed, the display of the list of resources will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case, and also presses the “Modify” button.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is currently in the process of modifying (or deleting) the same resource.
	4. The system will display a form that allows the information to be changed.
5. The user makes the necessary changes and presses “Submit”.	6. The system checks that the name of the resource was not changed to the name of another resource of the same type in the system.

7. The data is updated in the database.

8. The system displays a success/fail message to the user.

9. The display of the list of resources is refreshed if the name changed. This use case ends.

Alternative Courses

Line 2: User does not have permission to modify this particular type of resource. An error message is displayed and this use case ends.
Line 3: The system finds that another user is currently in the process of modifying the same resource. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the "Override Lock" button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the "OK" button is pressed instead, then this use case ends.
Line 5: The user presses "Cancel" instead of "Submit". In this case, the form, which allows the information to be changed, will close, and this use case ends.
Line 6: The name already exists for another resource of the same type in the system. An error message is displayed but the dialog box remains open. Return to line 5 of the typical course of events.
Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Delete Resource

Purpose

The purpose of this use case is to remove a resource from the system.

Overview

The admin-level user may need to remove a resource from the system. Resources will only be removable if they are not allocated to any projects (terminated or not). Usually, this use case will be performed if a client was added accidentally. It could also be used to purge information from the database (if it gets too large), but this would most likely not be done since it is desired to keep a history of all current and terminated projects. Maintaining this history will be difficult if the resources needed would not be available. This use case may only begin after the "Select Item" use case has been performed using a resource as the selected item. The user will select "Delete Resource" from the menu. The system will verify that the user actually has permission to perform this action. The system will check that the selected client is not allocated to any projects. The system will verify that no other user is in the process of modifying the selected resource. If it appears that another user is in the process of modifying the selected resource, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the "Override Lock" button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will prompt the user for confirmation. The system will remove the client from the system, display a success/fail message box, and then refresh the display of the list of clients (if the operation was successful).

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case using a resource as the item selected, and also selects “Delete Resource” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that this resource is not allocated to any projects (terminated or not).
	4. The system checks that no other user is in the process of modifying (or deleting) the same resource.
	5. System prompts for confirmation.
6. User presses the “Delete” button of the confirmation dialog box.	7. The system removes the resource from the database.
	8. The system displays a success/fail message box.
	9. The system refreshes the display.

Alternative Courses

Line 2: User does not have permission to delete this particular type of resource. An error message is displayed and this use case ends.
Line 3: The selected resource has projects allocated to it. An error message is displayed and this use case ends.
Line 4: The system finds that another user is currently in the process of modifying the same resource. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 5 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
Line 6: The user presses “Cancel” instead of “Delete”. The use case ends.
Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Add Project

Purpose

The purpose of this use case is to add a new project to the system.

Overview

The admin-level user needs to add another project to the system. The information that needs to be entered about the new project consists of: the name of the project, the client the project belongs to, and the status of the project. This use case begins when the admin-level user chooses the menu item for adding a project. The system will verify that the user actually has permission to perform this action. The system will display a dialog box that allows the user to enter the needed information. The user will be able to enter the information and press “Submit”, or the user will be able to press “Cancel” to abort the operation. After the user presses “Submit”, the system attempts to add the new project to the database and will display a message to indicate success or failure. The project name must be unique for the given client. If successful, the display of the list of projects will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses to add a project.	2. The system checks that the user is allowed to perform this operation.
	3. The system displays a dialog box that allows the user to enter the information about the project.
4. User provides the information and presses “Submit”.	5. Program checks to make sure the project name entered does not already exist in the system.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add a project. An error message is displayed and the use case ends.
- Line 4: User presses “Cancel” instead of “Submit”. The dialog box closes and the use case ends.
- Line 5: The project name already exists for given client. An error message is displayed but the dialog box remains open. Return to line 4 of the typical course of events.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Modify Project

Purpose

The purpose of this use case is to modify a project that already exists in the system.

Overview

The admin-level user needs to modify some of the information that the system keeps track of about a particular project. The information that may be modified consists of almost all the information originally entered in the “Add Project” use case: the project may undergo a name change, and the status of the project may change. The client may *not* be changed. This use case begins after the “Select Item” use case has been performed using a project as the selected item and the user will presses the “Modify” button to indicate the user’s desire to modify the selected project. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying the selected project. If it appears that another user is in the process of modifying the selected project, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will display a form (that resembles the “Add Project” form), which allows the information to be changed (of course, with the client name disabled, because the client name may not be changed). The user will be able to press “Submit” to accept the changes, or “Cancel” to abort the operation. If “Submit” was pressed, the system will check that the project name was not changed to a name that is already used by another project. The system will update the database. A message will be displayed to indicate success or failure. If the name was changed, the display of the list of projects will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case using a project as the selected item, and also presses the “Modify” button.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is currently in the process of modifying (or deleting) the same project.
	4. The system will display a form that allows the information to be changed.
5. The user makes the necessary changes and presses “Submit”.	6. The system checks that the name of the project was not changed to the name of another project in the system.
	7. The data is updated in the database.
	8. The system displays a success/fail message to the user.
	9. The display of the list of projects is refreshed if the name changed. This use case ends.

Alternative Courses

- Line 2: User does not have permission to modify a project. An error message is displayed and this use case ends.
- Line 3: The system finds that another user is currently in the process of modifying the same project. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
- Line 5: The user presses “Cancel” instead of “Submit”. In this case, the form, which allows the information to be changed, will close, and this use case ends.
- Line 6: The name already exists for another project in the system. An error message is displayed but the dialog box remains open. Return to line 5 of the typical course of events.
- Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Delete Project

Purpose

The purpose of this use case is to remove a project from the system.

Overview

The admin-level user may need to remove a project from the system or terminate a project. Projects will only be removable if they have no resource allocations allocated to them. If the do have resource allocations allocated to them, they will be terminated instead. Projects would be removed if they were added accidentally or if they did not get past the “Discovery” phase. The use case could also be used to purge information from the database (if it gets too large), but this would most likely not be done since it is desired to keep a history of all current and terminated projects. This use case may only begin after the “Select Item” use case has been performed using a project as the selected item and the user selects “Delete Project” from the menu. The system will verify that the user actually has permission to perform this action. The system will check whether or not the selected project has any resource allocations allocated to it. The system will verify that no other user is in the process of modifying (or deleting) the selected project. If it appears that another user is in the process of modifying the selected project, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will prompt the user for confirmation. This confirmation will indicate whether the project is being removed or terminated. The system will remove the project from the system (or terminate it), display a success/fail message box, and then refresh the display of the list of projects (if the project was removed).

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case using a project as the selected item, and also selects “Delete Project” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks the number of resource allocations allocated to the selected project.
	4. The system checks that no other user is in the process of modifying (or deleting) the same project.
	5. System prompts for confirmation. The prompt displays whether “remove” or “terminate” will be used.
6. User presses the “Delete” button of the confirmation dialog box.	7. The system removes the project from the database or puts it into the “terminated” state.
	8. The system displays a success/fail message box.
	9. The system refreshes the display if the project was removed.

Alternative Courses

- Line 2: User does not have permission to delete a project. An error message is displayed and this use case ends.
- Line 4: The system finds that another user is currently in the process of modifying the same project. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 5 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
- Line 6: The user presses “Cancel” instead of “Delete”. The use case ends.
- Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Add Resource Allocation

Purpose

The purpose of this use case is to add a new resource allocation to the system.

Overview

The admin-level user needs to add another resource allocation to the selected project. Before an allocation can be created, the project and the resource itself, that the resource allocation needs, already exist. A project lock must also be checked out. The project to add the resource allocation to must be selected in the tree before this use case can begin. The information that needs to be entered about the new resource allocation consists of: the type of resource to allocate and the actual resource to allocate. After the actual resource is selected, extra information about that resource will be shown in an available text field within that same dialog box. The use case begins when the admin-level user chooses the menu item for adding a resource allocation. The system will verify that the user actually has permission to perform this action. The system will display a dialog box that allows the user to enter the needed information. The user will be able to enter the information and press "Submit", or the user will be able to press "Cancel" to abort the operation. After the user presses "Submit", the system will attempt to add the new resource allocation to the project and add it to the database and will display a message to indicate success or failure. If successful, the display of the list of resource allocations will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the "Select Item" use case using a project as the selected item, and the user chooses "Add a resource allocation" from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks out a project lock and displays a dialog box that allows the user to enter the information about the resource allocation.
4. User provides the information and presses "Submit".	5. Program checks to make sure the resource allocation entered does not already exist in the system. There cannot be two (or more) resource allocations for the same project and same resource.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add a resource allocation. An error message is displayed and the use case ends.
- Line 4: User presses "Cancel" instead of "Submit". The dialog box closes and the use case ends.

- Line 5: The resource allocation already exists in the system. An error message is displayed but the dialog box remains open. Return to line 4 of the typical course of events.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Delete Resource Allocation

Purpose

The purpose of this use case is to remove a resource allocation from a given project.

Overview

The admin-level user may need to remove a resource allocation from a given project. Resource allocations will always be removable. However, there will be a prompt with a severe warning because deleting the resource allocation deletes all Timeslot records for that allocation. It is probably desired to keep these records for the purpose of maintaining a history. When removed, the system deletes all of the timeslots that are in this resource allocation. Then, it deletes the resource allocation itself. This use case may only begin after the “Select Project” use case has been performed. The user will select “Delete Resource Allocation” from the menu. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying the selected project (make sure the lock is not checked out). If it appears that another user is in the process of modifying the selected project, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will prompt the user for confirmation. The system will remove the resource allocation from the project (and all it’s timeslots), display a success/fail message box, and then refresh the display of the list of resource allocations for the given project (if the operation was successful).

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case using a project as the selected item, and also selects “Delete Resource Allocation” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is in the process of modifying (or deleting) the given project.
5. User presses the “Delete” button of the confirmation dialog box.	4. System prompts for confirmation.
	6. The system removes the resource allocation from the project and from the database. The system also removes all the resource allocation’s timeslots.

	7. The system displays a success/fail message box.
	8. The system refreshes the display.

Alternative Courses

- Line 2: User does not have permission to delete a resource allocation. An error message is displayed and this use case ends.
- Line 3: The system finds that another user is currently in the process of modifying the same project. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the "Override Lock" button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the "OK" button is pressed instead, then this use case ends.
- Line 5: The user presses "Cancel" instead of "Delete". The use case ends.
- Line 6: The system is unable to update the database. The system displays an error message and this use case ends.

Start GUI

Purpose

The purpose of this use case is to allow the user to start the GUI on the client machine and login as a user of the system. This use case must be performed first, since the user cannot interact with the system in any way until the GUI has been started.

Overview

This use case begins when the user starts the client application. The application attempts to connect to the remote server, and an error results if it is unable to do so. A form appears that prompts the user for the username and password. The user enters the information and clicks "OK". The system performs a lookup of the user and verifies the password. If the user is not found or if the password is incorrect, an error message will be displayed and the user will be allowed to retry. When the user is successful in entering login name and password, the system will display the main window of the GUI.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user runs the client application.	2. The system displays a login dialog box that allows the user to enter a login name and password.
3. The user enters the information and presses "OK".	4. The system finds the user in the database and verifies the password.
	5. The system displays the main window of the GUI. This use case ends.

Alternative Courses

- Line 3: Instead of pressing the "OK" button, the user closes the window. The application terminates and this use case ends.
- Line 4: The system cannot find a user in the database with the specified login name. An error message is displayed to the user, the form is cleared, and the user is allowed to retry. Return to line 3 in the typical course of events.
- Line 4: The system finds the users login name, but the password doesn't match. An error message is displayed to the user, the password field of the form is cleared, and the user is allowed to retry. Return to line 3 in the typical course of events.

Shutdown GUI

Purpose

The purpose of this use case is to allow the user to close the client application.

Overview

This use case begins when the user chooses "Exit" from the main menu of the GUI. The system determines whether or not the user was in the process of adding or modifying any information. If so, the system prompts the user for confirmation since some information that the user has entered might not be saved. Otherwise, the system removes all GUI windows and terminates the application.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses "Exit" from the main menu of the GUI.	2. The system displays a prompt asking the user for confirmation since the user was in the process of adding/modifying some information.
3. The user presses "Continue".	4. The system removes all GUI windows and terminates the application. This use case ends.

Alternative Courses

- Line 2: The user was not in the process of adding/modifying any information. In this case, proceed to line 4 of the typical course of events since a confirmation dialog is not needed.
- Line 3: The user presses "Cancel" instead of "Continue". Close the confirmation dialog. This use case ends.

Add User

Purpose

The purpose of this use case is to add a new user to the system.

Overview

The admin-level user needs to add another user to the system because the user is new to the system and has never logged on before. The information that needs to be entered about the new user consists of: name, username, initial password and the access level of the user. The admin-level user will choose the menu item for adding a user. The system will verify that the user actually has permission to perform this action. The system will display a dialog box that allows the user to enter the needed information. The default access level of the user will be view-only. The user will be able to enter the information and press “Submit”, or the user will be able to press “Cancel” to abort the operation. After the user presses “Submit”, the system will attempt to add the new user to the database and will display a message to indicate success or failure.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses to add a user.	2. The system checks that the user is allowed to perform this operation.
	3. The system displays a dialog box that allows the user to enter the information about the user.
4. User provides the information and presses “Submit”.	5. The system checks to make sure the username entered does not already exist in the system.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add a user. An error message is displayed and the use case ends.
- Line 4: User presses “Cancel” instead of “Submit”. The dialog box closes and the use case ends.
- Line 5: The username already exists for another user in the system. An error message is displayed but the dialog box remains open. Return to line 4 of the typical course of events.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Modify User

Purpose

The purpose of this use case is to modify a user that already exists in the system.

Overview

The admin-level user needs to modify some of the information that the system keeps track of about a particular user. The information that may be modified consists of any information originally entered in the “Add User” use case: the user may undergo a name change, the username may undergo a change, the user’s password may have been forgotten and needs to be reset or the user has been promoted and the access level needs to be changed. This use case begins when the user chooses the “Modify User” menu item. The system will verify that the user actually has permission to perform this action. The system will open a modify user dialog, which will allow the user to select a particular user to modify by choosing the name in a selection list. The system will then fill in the information on the selected user in the rest of the dialog. The user will then make any necessary changes to the information in the dialog. The user will be able to press “Submit” to accept the changes, or “Cancel” to abort the operation. If “Submit” was pressed, the system will check that the username was not changed to a username that is already used by another user. The system will update the database. A message will be displayed to indicate success or failure.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user selects the “Modify User” menu item.	2. The system checks that the user is allowed to perform this operation.
	3. The system will display a dialog that allows the user to select a user to modify.
4. The user will select a user to modify by selecting the persons name in the drop selection list.	5. The system will fill in the remaining portion of the dialog with the selected user information.
6. The user makes the necessary changes and presses “Submit”.	7. The system checks that the username of the user was not changed to the username of another user in the system.
	8. The data is updated in the database.
	9. The system displays a success/fail, message to the user and the use case ends.

Alternative Courses

Line 2: User does not have permission to modify a user. An error message is displayed and this use case ends.
Line 6: The user presses “Cancel” instead of “Submit”. In this case, the form, which allows the information to be changed, will close, and this use case ends.
Line 7: The username already exists for another user in the system. An error message is displayed but the dialog box remains open. Return to line 6 of the typical course of events.
Line 8: The system is unable to update the database. The system displays an error message and this use case ends.

Delete User

Purpose

The purpose of this use case is to remove a user from the system.

Overview

The admin-level user may need to remove a user from the system. Usually, this use case will be performed if a user no longer works for the company or no longer needs access to the program. The user will select “Delete User” from the menu. The system will verify that the user actually has permission to perform this action. The system will open a dialog box that will allow the user to select from a list of users to choose the user to delete. The system will verify that the user is not attempting to delete his/her self from the system. The system will prompt the user for confirmation. The system will remove the user from the system, and then display a success/fail message box.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user selects “Delete User” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system will display a list of users to choose from.
4. The user selects the user to delete from the selection list.	5. They system verifies that the user is not attempting to remove his/her own self from the system.
	6. The system prompts for confirmation.
7. User presses the “Delete” button of the confirmation dialog box.	8. The system removes the user from the database .
	9. The system displays a success/fail message box and the use case ends.

Alternative Courses

- Line 2: User does not have permission to delete a user. An error message is displayed and this use case ends.
- Line 5: The system determines that the user is attempting to remove him/her own self from the system. The system displays an error message box and the use case ends.
- Line 7: The user presses “Cancel” instead of “Delete”. The use case ends.
- Line 8: The system is unable to update the database. The system displays an error message and this use case ends.

Change Password

The use case begins when a user chooses to change their own password. Prompt and verify and prompt for the new password twice. Check and update the database.

Purpose

The purpose of this use case is to allow any user to modify his/her own password.

Overview

This use case is needed to allow a user to change the user's password. This use case would be used after the user logs in for the first time, in order to replace the password that the admin-level user assigned when using the "Add User" use case. It would also be used later, as needed by the user. This use case begins when the user selects "Change My Password" from the main menu. The system responds by displaying a dialog box that lets the user enter the old password, the new password, and a confirmation of the new password. The user may enter the information and press "OK", or may close the window to abort this use case. The system verifies the user and displays an error if not all the information was correct, in which case the user may retry. Once correct information is entered, the system displays the main window of the GUI.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user selects "Change My Password" from the main menu.	2. The system displays a dialog box that allows the user to enter the old password, the new password, and the confirmation of the new password.
3. The user enters the information and presses "OK".	4. The system verifies the information.
	5. The system updates the information in the database.
	6. The system closes the dialog box, displays a success/fail message, and this use case ends.

Alternative Courses

- Line 3: The user presses "Cancel" instead of "OK". Close the dialog box. This use case ends.
- Line 4: The system determines that either: the old password is wrong, the new password matches the old password, or the new password is not correctly confirmed. In any case, the system displays an error message, clears the form, and allows the user to retry. Return to line 3 of the typical course of events.
- Line 5: The system is unable to update the database. The system displays an error message and this use case ends.

Forecast Revenue

Purpose

The purpose of this use case is to allow the admin-level user to generate a report that predicts total revenue, cost, and profit for a specified time period.

Overview

Only admin-level users may execute this use case. The user can choose to forecast revenue for a single project, or for all projects. The results will be written to a file in HTML format and the application will launch a web-browser that points at the generated file, in order to display the results. This use case may begin in one of two ways:

The user may choose "Forecast Revenue Across All Projects" from the menu.
The user may use the "Select Item" use case, choosing a project as the selected item. The user would then choose "Forecast Revenue for the Selected Project" from the menu.

The system will respond by generating the report and launching a web-browser that points at the file's URL. If the system is unable to launch the web-browser, an error message box will be displayed which will display the target URL to the user so that the user can launch the web-browser manually.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses "Forecast Revenue Across All Projects" from the menu.	2. The system verifies that the user has permission to perform this function.
	3. The system generates the report by accessing the database and performing the needed calculations.
	4. The system creates an HTML file and writes it to disk where it can be accessed by a web-browser.
	5. The system launches a web-browser that points at the file's URL. This use case ends.

Alternative Courses

- Line 1: The user can instead: Use the "Select Item" use case, choosing a project as the selected item. The user would then choose "Forecast Revenue for the Selected Project" from the menu. Proceed to line 2 of the typical course of events.
- Line 2: The user does not have permission to perform this function. The system displays an error message and this use case ends.
- Line 3: The system is unable to access the database. The system displays an error message and this use case ends.
- Line 4: The file cannot be written to disk. The system displays an error message and this use case ends.

- Line 5: The system is unable to launch an external web-browser. The system displays an error message and also displays the URL of the file so that the user can manually launch a web-browser. This use case ends.

Generate Staffing Report

Purpose

The purpose of this use case is to allow the admin-level user to generate a staffing report for all Consultants in the system.

Overview

Only admin-level users may execute this use case. The staffing report, generated by this use case, will have a section for each consultant. It will list the projects that the consultant has worked on during the specified time interval. It will show the consultant details such as name, e-mail, rate/cost, etc. For each project listed, it will show the start date and end date, the total number of hours that the consultant is scheduled to work for that project, and the forecasted revenue for that project. This use case begins when the admin-user chooses "Generate Staffing Report" from the main menu. The system will respond by generating the report and launching a web-browser that points at the file's URL. If the system is unable to launch the web-browser, an error message box will be displayed which will display the target URL to the user so that the user can launch the web-browser manually.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses "Generate Staffing Report" from the menu.	2. The system verifies that the user has permission to perform this function.
	3. The system generates the report by accessing the database and performing the needed calculations.
	4. The system creates an HTML file and writes it to disk where it can be accessed by a web-browser.
	5. The system launches a web-browser that points at the file's URL. This use case ends.

Alternative Courses

- Line 3: The system is unable to access the database. The system displays an error message and this use case ends.
- Line 2: The user does not have permission to perform this function. The system displays an error message and this use case ends.
- Line 4: The file cannot be written to disk. The system displays an error message and this use case ends.

- Line 5: The system is unable to launch an external web-browser. The system displays an error message and also displays the URL of the file so that the user can manually launch a web-browser. This use case ends.

Generate Conflict Report

Purpose

The purpose of this use case is to allow the admin-level user or the project-lead-level user to generate a conflict report.

Overview

View-only users may not execute this use case. The conflict report, generated by this use case, will have a section for each resource. In each section, the details of all conflicts will be listed. When adding timeslots, conflicts are detected and displayed to the user as warnings (which can be ignored). This use case is the only way to check for conflicts after the initial warning message is dismissed. This use case begins when the admin-user chooses "Generate Conflict Report" from the main menu. The system will respond by generating the report and launching a web-browser that points at the file's URL. If the system is unable to launch the web-browser, an error message box will be displayed which will display the target URL to the user so that the user can launch the web-browser manually.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user chooses "Generate Conflict Report" from the menu.	2. The system verifies that the user has permission to perform this function.
	3. The system generates the report by accessing the database and performing the needed calculations.
	4. The system creates an HTML file and writes it to disk where it can be accessed by a web-browser.
	5. The system launches a web-browser that points at the file's URL. This use case ends.

Alternative Courses

- Line 2: The user does not have permission to perform this function. The system displays an error message and this use case ends.
- Line 3: The system is unable to access the database. The system displays an error message and this use case ends.
- Line 4: The file cannot be written to disk. The system displays an error message and this use case ends.

- Line 5: The system is unable to launch an external web-browser. The system displays an error message and also displays the URL of the file so that the user can manually launch a web-browser. This use case ends.

Add Timeslot

Purpose

The purpose of this use case is to add a new timeslot to a given resource allocation.

Overview

The admin-level or project lead-level user needs to add another timeslot to the selected resource allocation. A *project* lock must be checked out for this operation. The resource allocation to add the timeslot to must be selected in the tree before this use case can begin. The information that needs to be entered about the new timeslot consists of: start date, stop date, hours-per-day, target rate, target overtime rate, and office. This use case begins when the admin-level or project lead-level user uses the “Select Item” use case using a resource allocation as the selected item and chooses the “Add Timeslot” menu item. The system will verify that the user actually has permission to perform this action. The system will display dialog box that allows the user to enter the needed information. The user will be able to enter the information and press “Submit”, or the user will be able to press “Cancel” to abort the operation. After the user presses “Submit”, the system will attempt to add the new timeslot to the resource allocation and add it to the database and will display a message to indicate success or failure. While trying to add the timeslot to a resource allocation, the system will check for conflicts with other timeslots. The possible conflicts are: “Location Conflict”, “Different Project, Same Location Conflict”, and “Same Project, Same Location Conflict”. If there are any conflicts, the user will be presented with an error message box describing the situation and asking if the user wishes to continue. If the user continues, the system will add the timeslot. If successful, the display of the list of timeslots will be refreshed, and graphs will be updated as needed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user first selects a resource allocation, and then chooses to add a timeslot.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks out a *project* lock and displays a dialog box that allows the user to enter the information about the timeslot.
4. User provides the information and presses “Submit”.	5. Check to see if the new timeslot conflicts with any other existing timeslots.
	6. The system stores the data in the database.
	7. The system closes the dialog box, displays a message to indicate that the operation was successful, and this use case ends.

Alternative Courses

- Line 2: User does not have permission to add a timeslot. An error message is displayed and the use case ends.
- Line 4: User presses "Cancel" instead of "Submit". The dialog box closes and the use case ends.
- Line 5: The timeslot conflicts with another one in the system. A warning message is displayed. The user may still add the timeslot if they choose. If they press "Cancel" the use case will end.
- Line 6: The system is unable to access the database. The system displays an error message and the use case ends.

Modify Timeslot

Purpose

The purpose of this use case is to modify a timeslot that exists in the system.

Overview

The admin-level or project lead-level user needs to modify some of the information that the system keeps track of about a particular timeslot for a given resource allocation. The information that may be modified consists of any information originally entered in the "Add Timeslot" use case. This use case may only begin after the "Select Item" use case has been performed using a timeslot as the selected item, and the user presses the "Modify" button. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying (or deleting) the project of the selected timeslot. If it appears that this is the case, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the "Override Lock" button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will display a form (that resembles the "Add Timeslot" form), which allows the information to be changed. The user will be able to press "Submit" to accept the changes, or "Cancel" to abort the operation. If "Submit" was pressed, the system will check for any possible conflicts. Conflict checking works the same way that it does for the "Add Timeslot" use case (except that the timeslot being modified must be explicitly excluded from the algorithm). A message will be displayed to indicate success or failure. If the name was changed, the display of the list of timeslots will be refreshed.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the "Select Item" use case using a timeslot as the selected item, and also presses the "Modify" button.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is currently in the process of modifying (or deleting) the same *project*.
	4. The system will display a form that allows

	the information to be changed.
5. The user makes the necessary changes and presses “Submit”.	6. The system checks if the timeslot has any conflicts.
	7. The data is updated in the database.
	8. The system displays a success/fail message to the user.
	9. The display of the list of timeslots is refreshed if the timeslot was modified. This use case ends.

Alternative Courses

- Line 2: User does not have permission to modify a timeslot. An error message is displayed and this use case ends.
- Line 3: The system finds that another user is currently in the process of modifying the same project. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
- Line 5: The user presses “Cancel” instead of “Submit”. In this case, the form, which allows the information to be changed, will close, and this use case ends.
- Line 6: The timeslot conflicts with another one in the system. A warning message is displayed. The user may still add the timeslot if they choose. If they press “Cancel” the use case will end.
- Line 7: The system is unable to update the database. The system displays an error message and this use case ends.

Delete Timeslot

Purpose

The purpose of this use case is to delete a timeslot that exists in the system.

Overview

The admin-level or project lead-level user may need to remove a timeslot from a given resource allocation. Timeslots will always be removable. This use case may only begin after the “Select Item” use case has been performed using a timeslot as the selected item, and the user selects “Delete Timeslot” from the menu. The system will verify that the user actually has permission to perform this action. The system will verify that no other user is in the process of modifying the selected *project* (make sure the lock is not checked out). If it appears that another user is in the process of modifying the selected project, a message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. The system will prompt the user for confirmation. The system will remove the timeslot from the resource allocation, display a success/fail message box, and then refresh the display of the list of timeslots for the given resource allocation (if the operation was successful).

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user performs the “Select Item” use case using a timeslot as the selected item, and also selects “Delete Timeslot” from the menu.	2. The system checks that the user is allowed to perform this operation.
	3. The system checks that no other user is in the process of modifying (or deleting) the given *project*.
	4. System prompts for confirmation.
5. User presses the “Delete” button of the confirmation dialog box.	6. The system removes the timeslot from the project and from the database.
	7. The system displays a success/fail message box.
	8. The system refreshes the display.

Alternative Courses

- Line 2: User does not have permission to delete a timeslot. An error message is displayed and this use case ends.
- Line 3: The system finds that another user is currently in the process of modifying the same project. A message will be displayed giving the name of the other user and the time that the other user began modification. The user will be able to by-pass the warning by pressing the “Override Lock” button, in the event that an error occurred and the other user is not actually trying to modify the information. If this button is pressed, proceed to line 4 of the typical course of events. If the “OK” button is pressed instead, then this use case ends.
- Line 5: The user presses “Cancel” instead of “Delete”. The use case ends.
- Line 6: The system is unable to update the database. The system displays an error message and this use case ends.

Display Project-View Graph

Purpose

The purpose of this use case is to draw the project view graph.

Overview

The user needs to view a particular project’s resource allocations and timeslots in a graphical layout. The graph for the particular project needs to be generated and displayed in the proper location in the GUI. This use case begins when the user either selects a different project, changes the graph selection tab from to project view, or changes the start and end date for the graph generation. The system will then use the last project selected using the "Select Item" use case to generate a graph that contains all the resource allocations and timeslots for that project and display it to the screen.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the "Select Item" use case is used and the item selected is a project.	2. The system determines the last selected project.
	3. The system generates the graph in regards to the project status level and the start and end date. The use case ends.

Alternative Courses

- Line 1: The user selects the project view tab for the graph pane of the main GUI. Proceed to line 2 of the typical course of events.
- Line 1: The user changed the start or end date for the graph generation. Proceed to line 2 of the typical course of events.

Display Resource-View Graph

Purpose

The purpose of this use case is to draw a resource view graph.

Overview

The user needs to view a particular resource type's resources and their resource allocations and timeslots in a graphical layout. The graph for the particular resource type needs to be generated and displayed in the proper location in the GUI. This location will be the panel in the upper-right of the main window. This use case begins when the user either selects a different resource type graph selection or changes the start and end date for the graph. The system will respond by generating the graph for the particular resource type and displaying it on the screen.

Typical Course of Events

Actor Action	System Response
1. This use case begins when the user selects a different resource type graph tab.	2. The system generates the graph in regards to the resource type and the start and end date for the graph. The use case ends.

Alternative Courses

- Line 1: The user changes the start or end date for the graph generation. Proceed to line 2 of the typical course of events.
-

Use Case Diagrams

Use case diagrams are constructed after all the use cases were identified and written. Use case diagrams serve several purposes. First, they allow all the use cases to be viewed at a glance. Second, they make it easy to identify the external actors (users) and how they use the system. Third and most importantly, they show which actors can participate in each use case (Larman 55).

In the system being designed, there are three different actors. Each actor corresponds to a different access level in the system. Only AdminUsers are allowed to participate in all use cases; the other two user types are limited in what they can do for security reasons. Three different use case diagrams were constructed to clearly show the difference between the access levels.

Figure 1: Use Case Diagram Showing Use Cases Only Available To AdminUsers

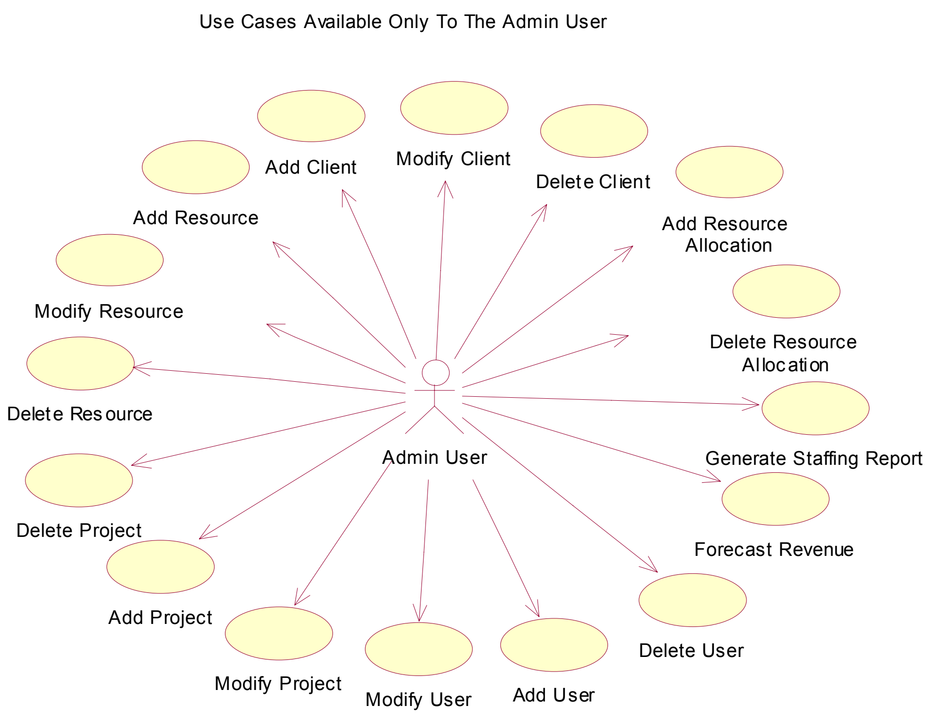


Figure 2: Use Case Diagram Showing Uses Cases Available to Both AdminUsers and ProjectLeadUsers

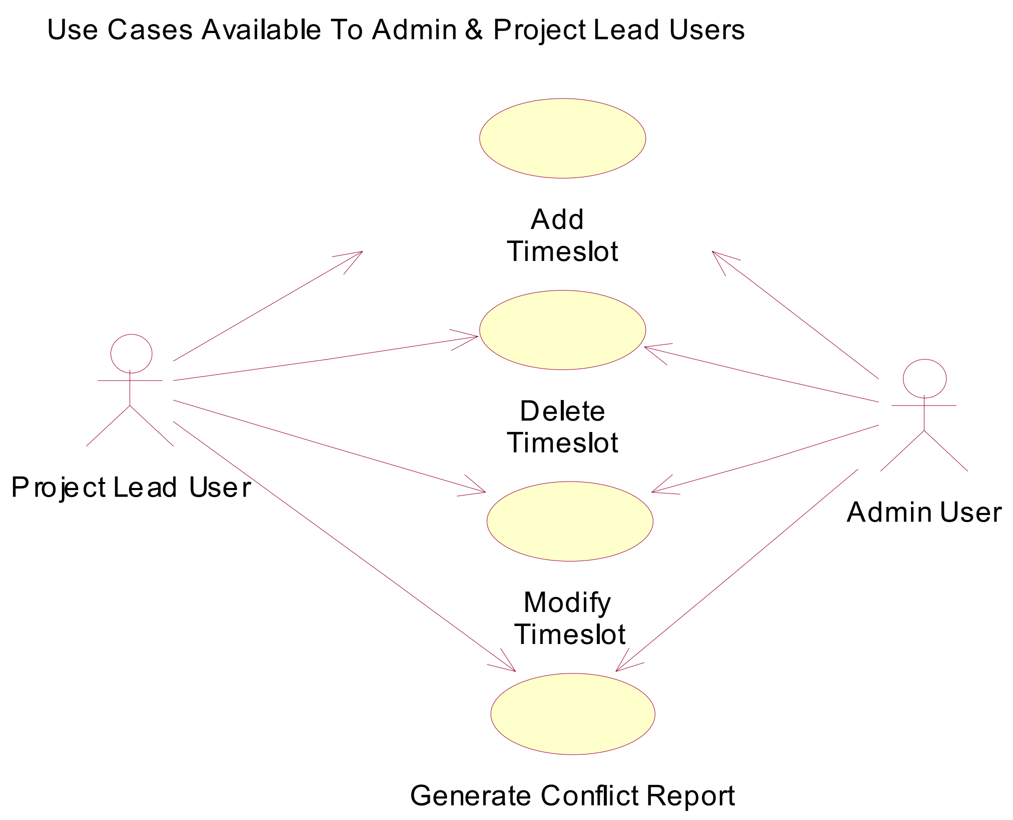
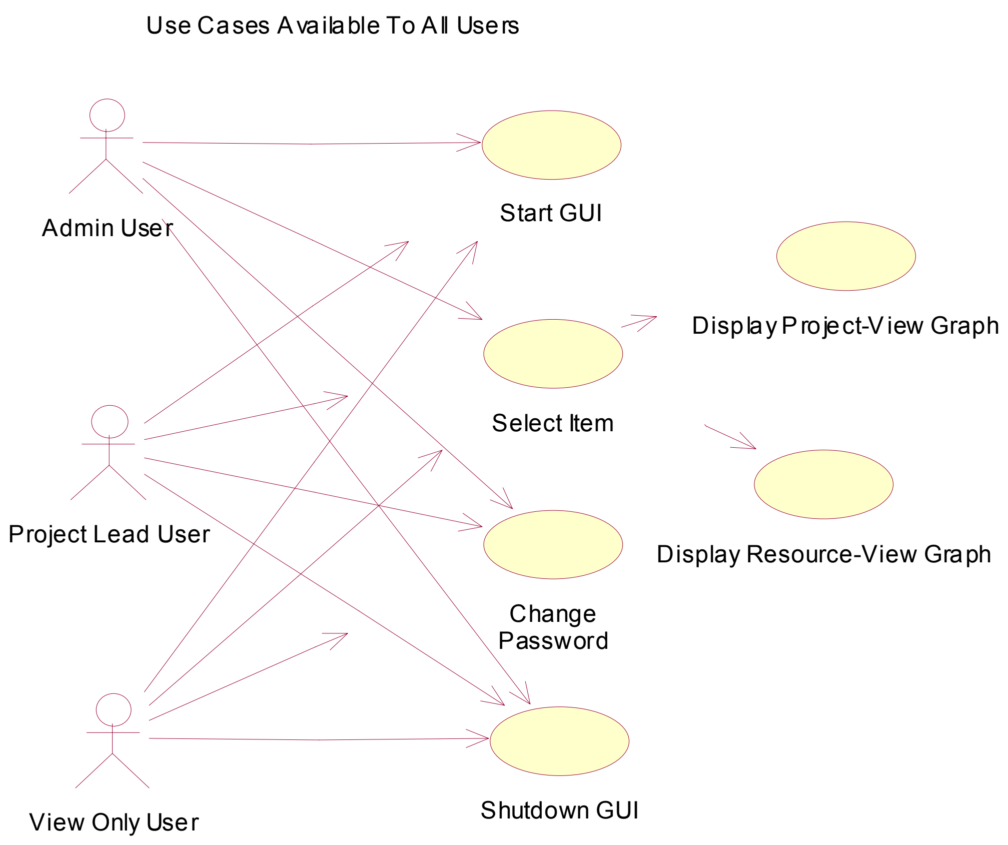


Figure 3: Use Case Diagram Showing Use Cases Available To All Users

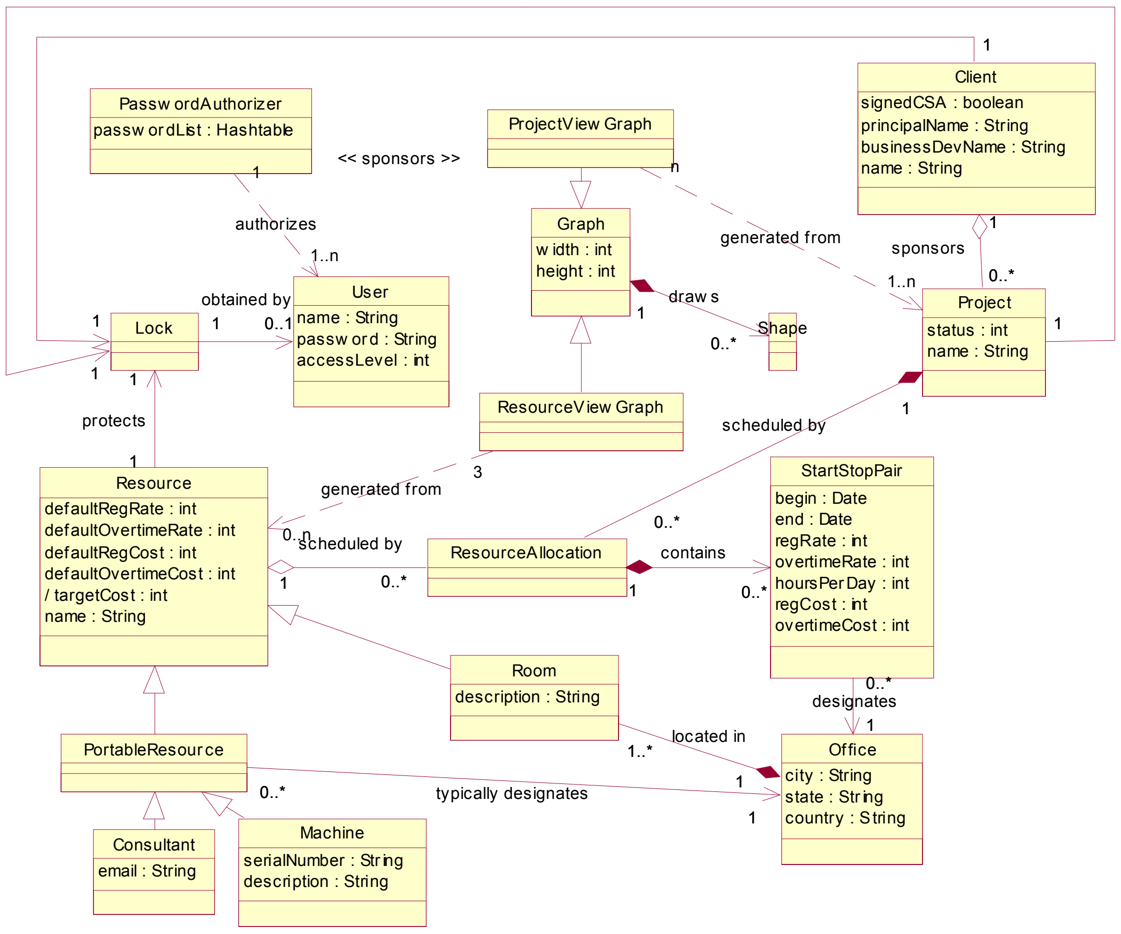


Conceptual Model

Conceptual models focus on identifying the domain elements and their attributes. No operations (methods) are defined on the conceptual model (Larman 87).

The conceptual model below defines many entity classes. Entity classes are data-rich classes that usually represent real objects in the problem domain. These entity classes will also be the ones stored in the database. Some entity classes listed in the conceptual model below include Client, Project, Resource, and Resource Allocation. Some service classes, such as PasswordAuthorizer and Lock are also identified in the conceptual model. The conceptual model lacks detailed descriptions of important classes related to the graphical user interface (GUI) and database access.

Figure 4: Conceptual Model



Sequence Diagrams

Sequence diagrams illustrate user interactions with the system and the system operations initiated by them. They show the order of operations required to complete a task. They are especially useful for determining what methods are needed in a class (Larman 137).

Sequence diagrams come in two forms. High-level sequence diagrams give an overall picture of the classes needed in the system. Some of the details, such as database access and GUI events are abstracted away. These high-level sequence diagrams are also useful for determining if the general division of operations between classes. Low-level sequence diagrams show all of the steps (function calls) needed to perform an operation. All participating classes are shown. Low level sequence diagrams allow the methods (including method signatures) needed in each class to be discovered.

In the design of this system, high-level sequence diagrams were created for each use case written. Low-level sequence diagrams were then constructed for tasks considered to be especially important. The sequence diagrams constructed can be found below. All of the diagrams are high-level unless otherwise noted.

Figure 5: Sequence Diagram for Add Client Use Case

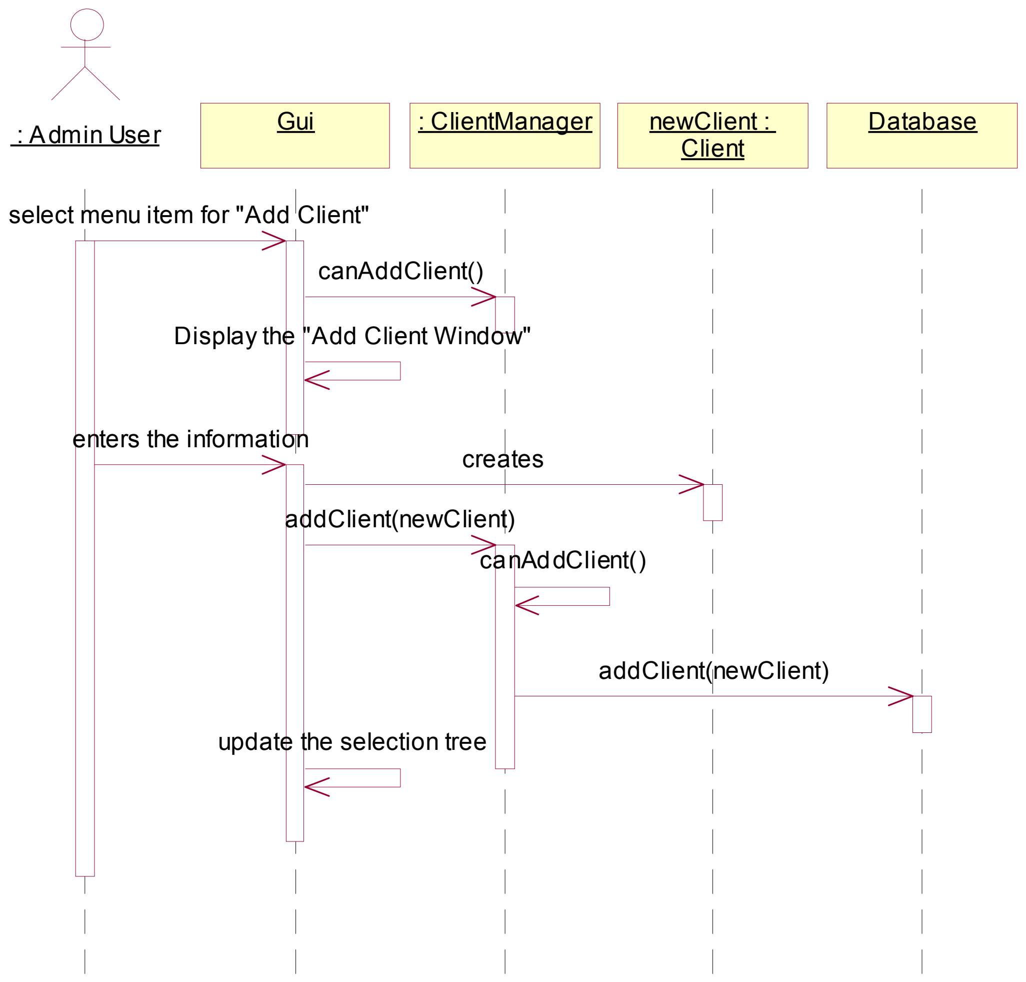


Figure 6: Sequence Diagram for Modify Client Use Case

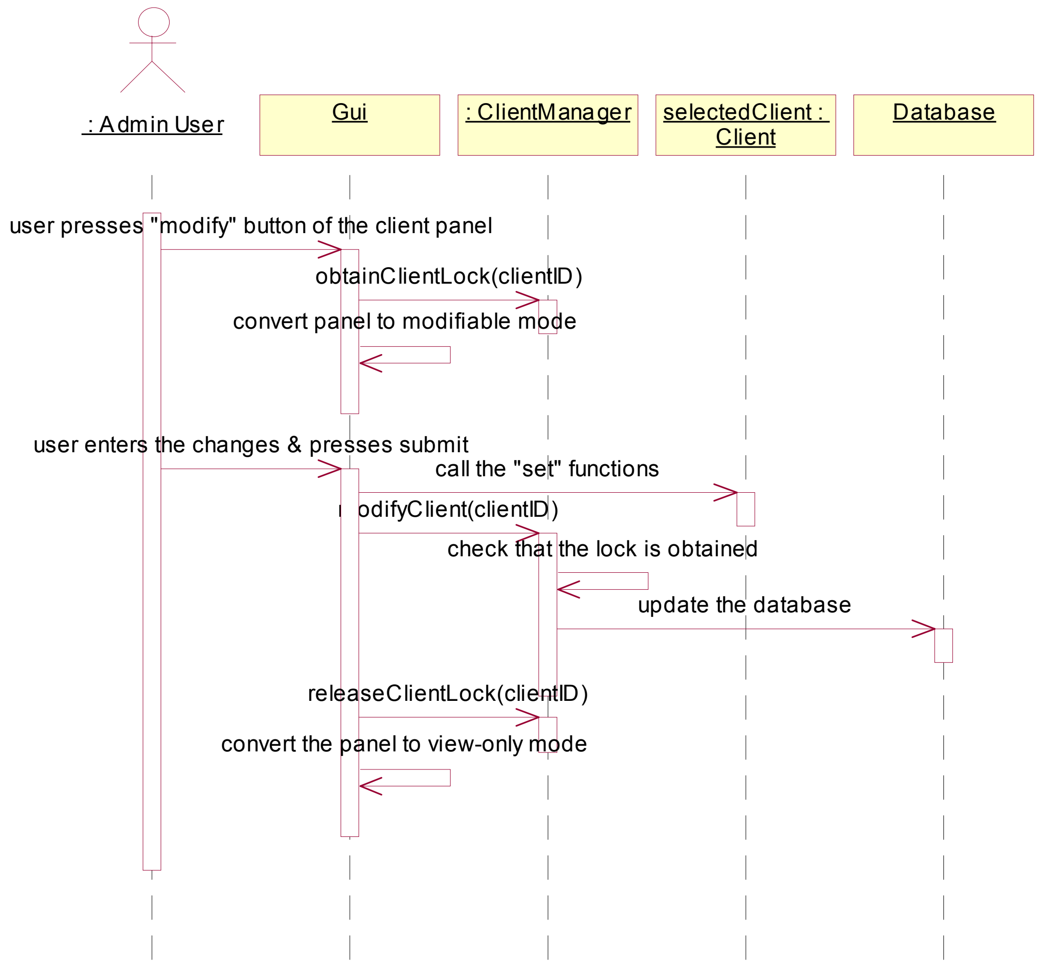

: Admin User
Gui
: ClientManager
selectedClient : Client
Database
user presses "modify" button of the client panel
obtainClientLock(clientID)
convert panel to modifiable mode
user enters the changes & presses submit
call the "set" functions
modifyClient(clientID)
check that the lock is obtained
update the database
releaseClientLock(clientID)
convert the panel to view-only mode

Figure 7: Sequence Diagram for Delete Client Use Case (No Projects Assigned to Client)

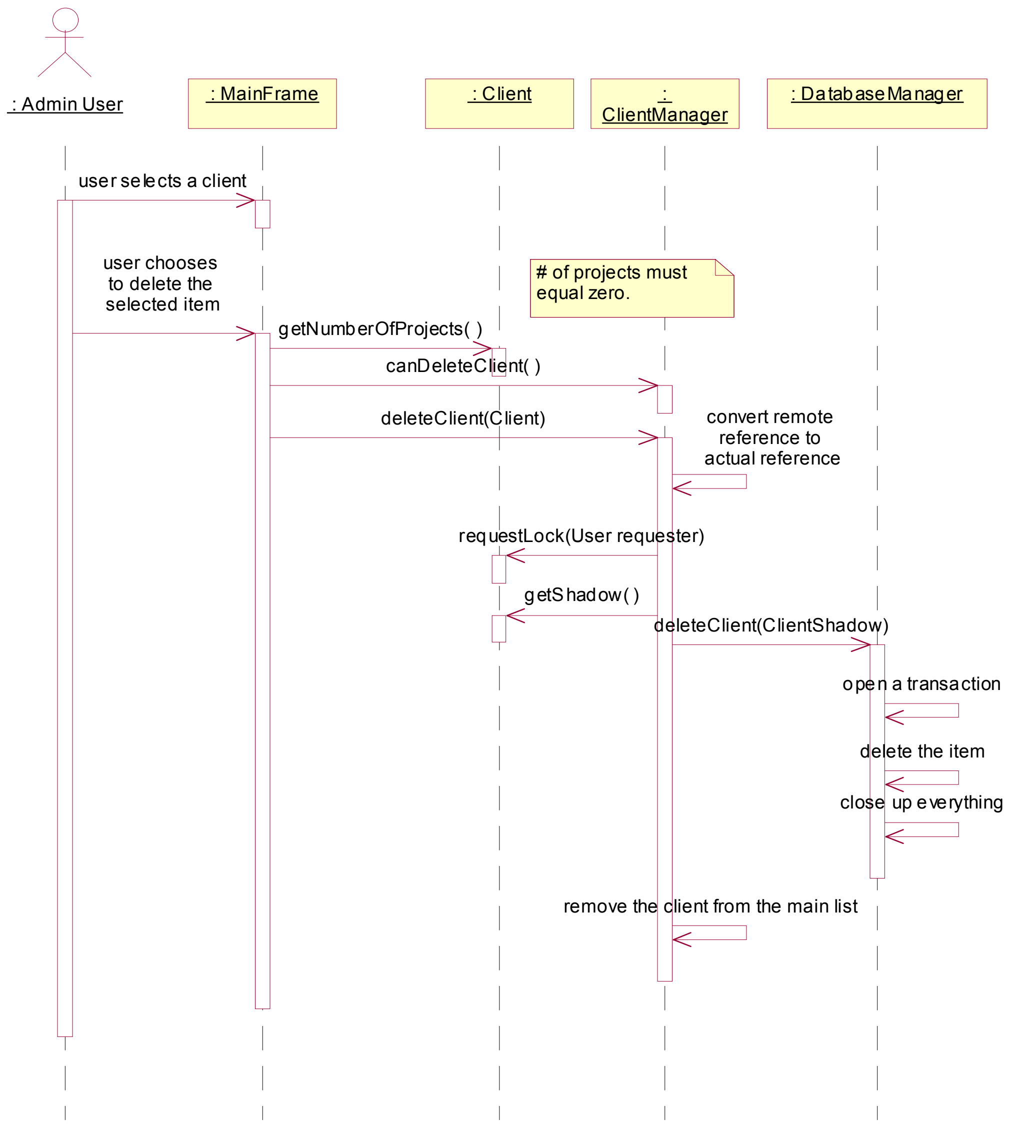


Figure 8: Sequence Diagram for Delete Client Use Case (Some Projects Assigned to Client)

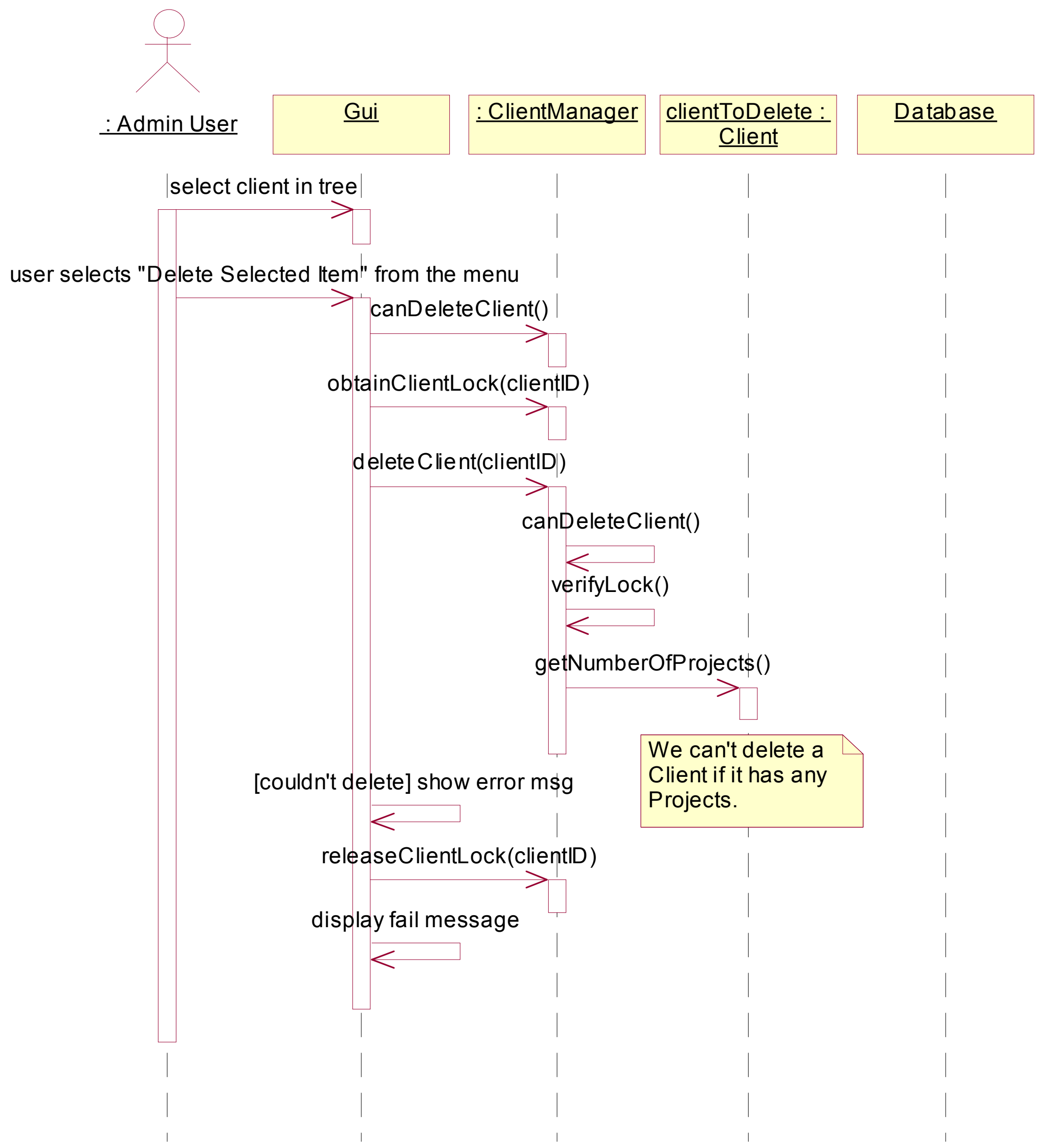


Figure 9: Sequence Diagram for Add Project Use Case

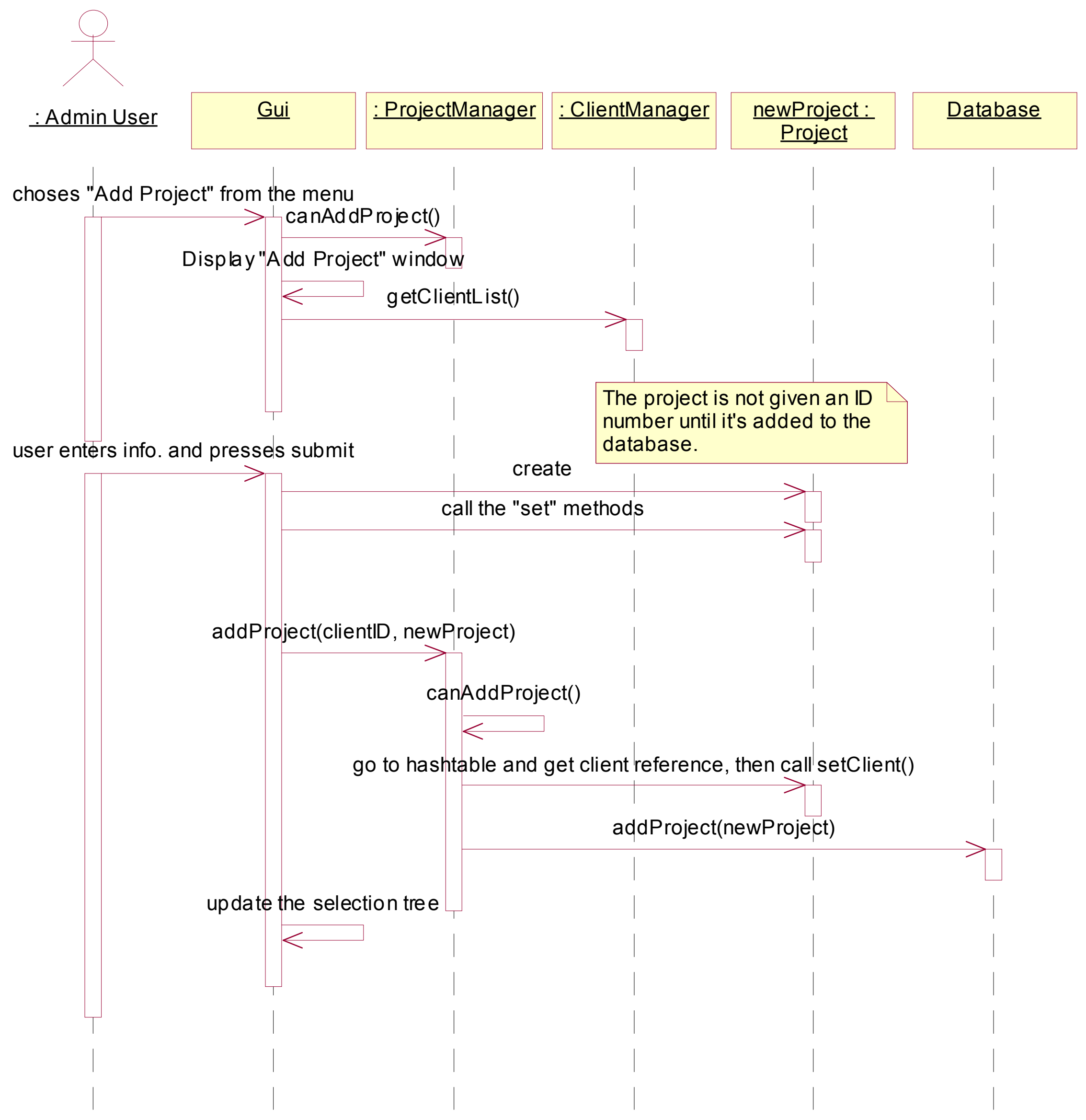

: Admin User
Gui
: ProjectManager
: ClientManager
newProject : Project
Database
choses "Add Project" from the menu
canAddProject()
Display "Add Project" window
getClientList()
The project is not given an ID number until it's added to the database.
user enters info. and presses submit
create
call the "set" methods
addProject(clientID, newProject)
canAddProject()
go to hashtable and get client reference, then call setClient()
addProject(newProject)
update the selection tree

Figure 10: Sequence Diagram for Modify Project Use Case

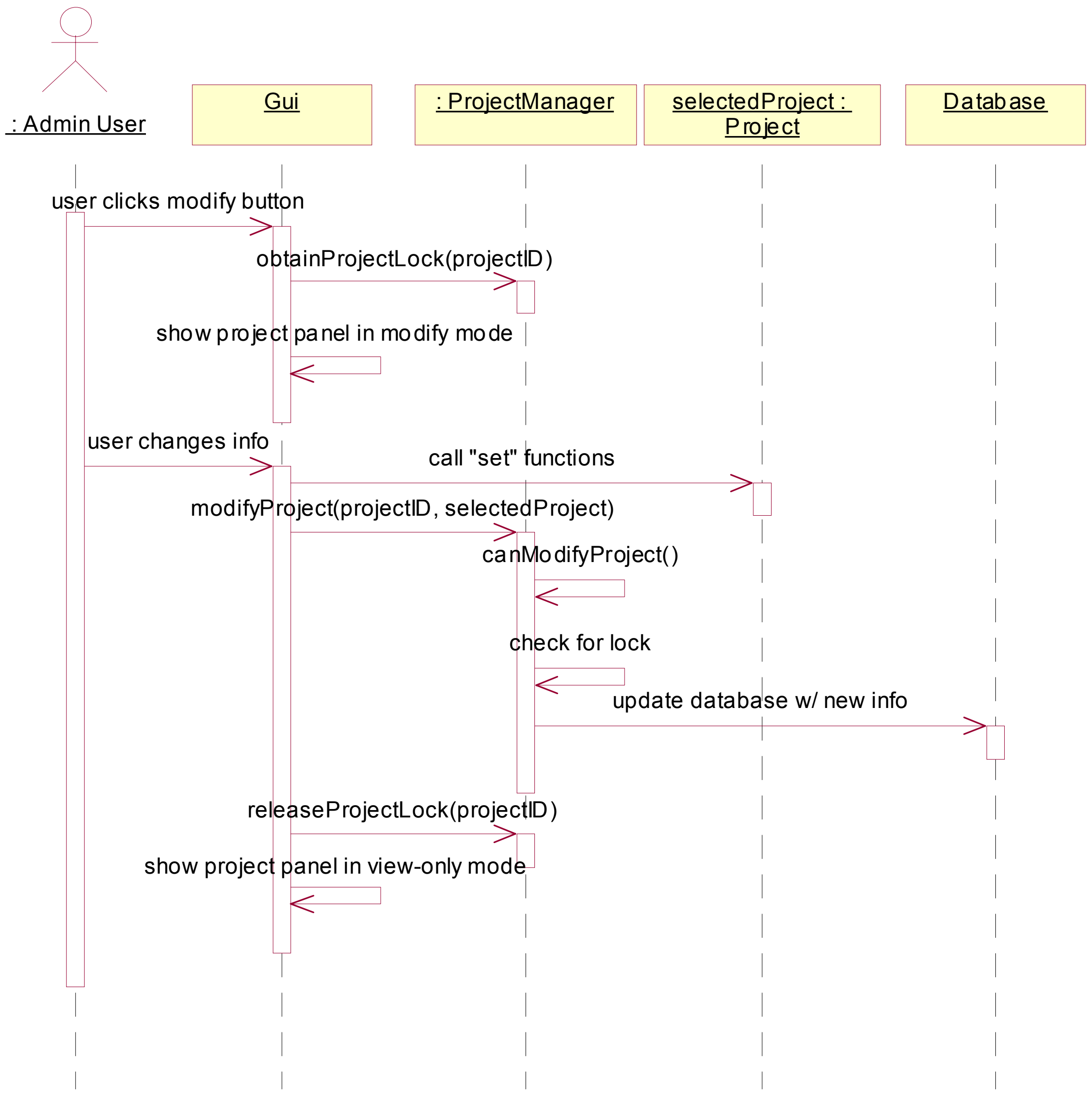

: Admin User
Gui
: ProjectManager
selectedProject : Project
Database
user clicks modify button
obtainProjectLock(projectID)
show project panel in modify mode
user changes info
call "set" functions
modifyProject(projectID, selectedProject)
canModifyProject()
check for lock
update database w/ new info
releaseProjectLock(projectID)
show project panel in view-only mode

Figure 11: Sequence Diagram for Delete Project Use Case (No Resource Allocations)

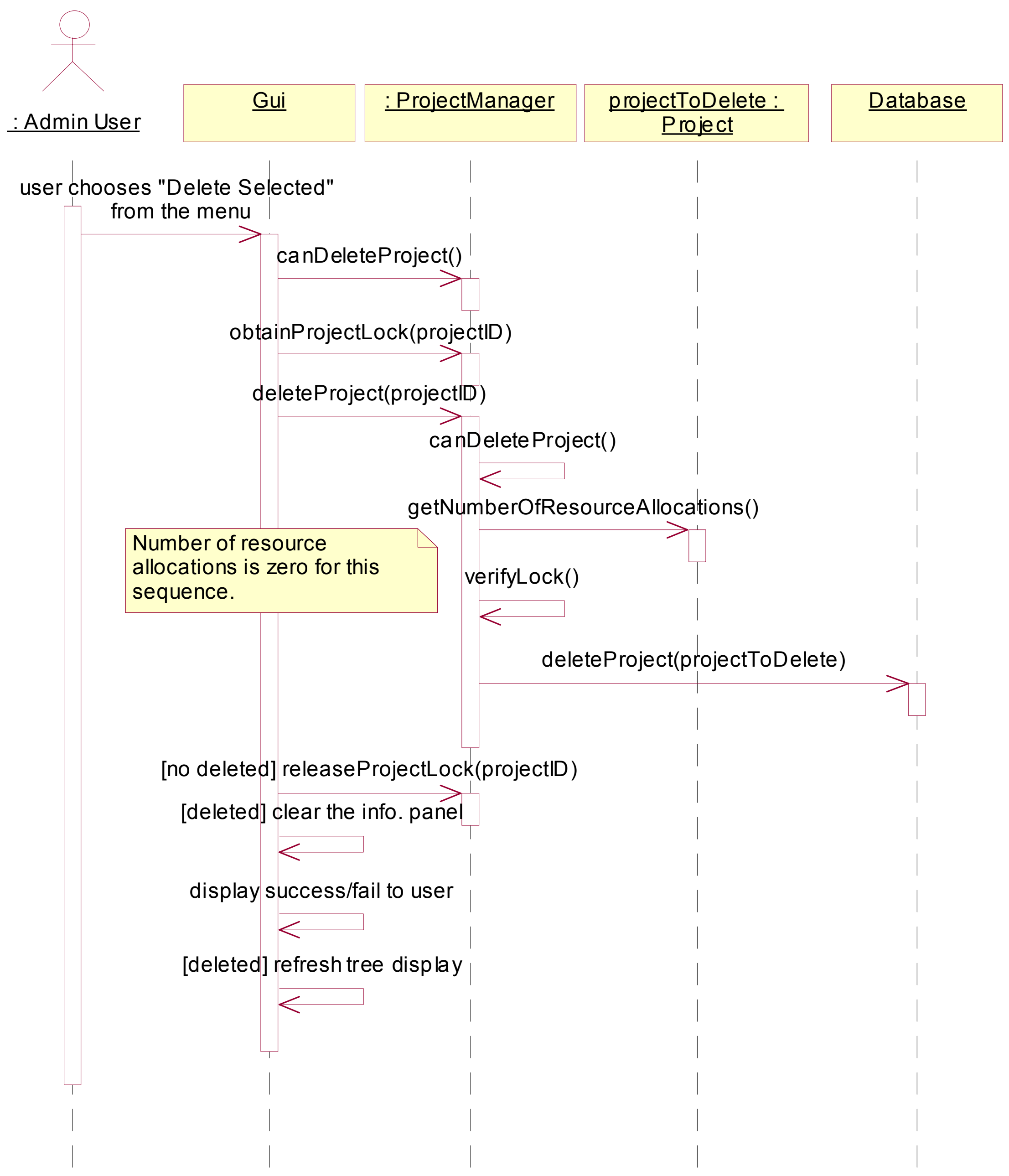

: Admin User
Gui
: ProjectManager
projectToDelete : Project
Database
user chooses "Delete Selected" from the menu
canDeleteProject()
obtainProjectLock(projectID)
deleteProject(projectID)
canDeleteProject()
getNumberOfResourceAllocations()
Number of resource allocations is zero for this sequence.
verifyLock()
deleteProject(projectToDelete)
[no deleted] releaseProjectLock(projectID)
[deleted] clear the info. panel
display success/fail to user
[deleted] refresh tree display

Figure 12: Sequence Diagram for Delete Project Use Case (Some Resource Allocations)

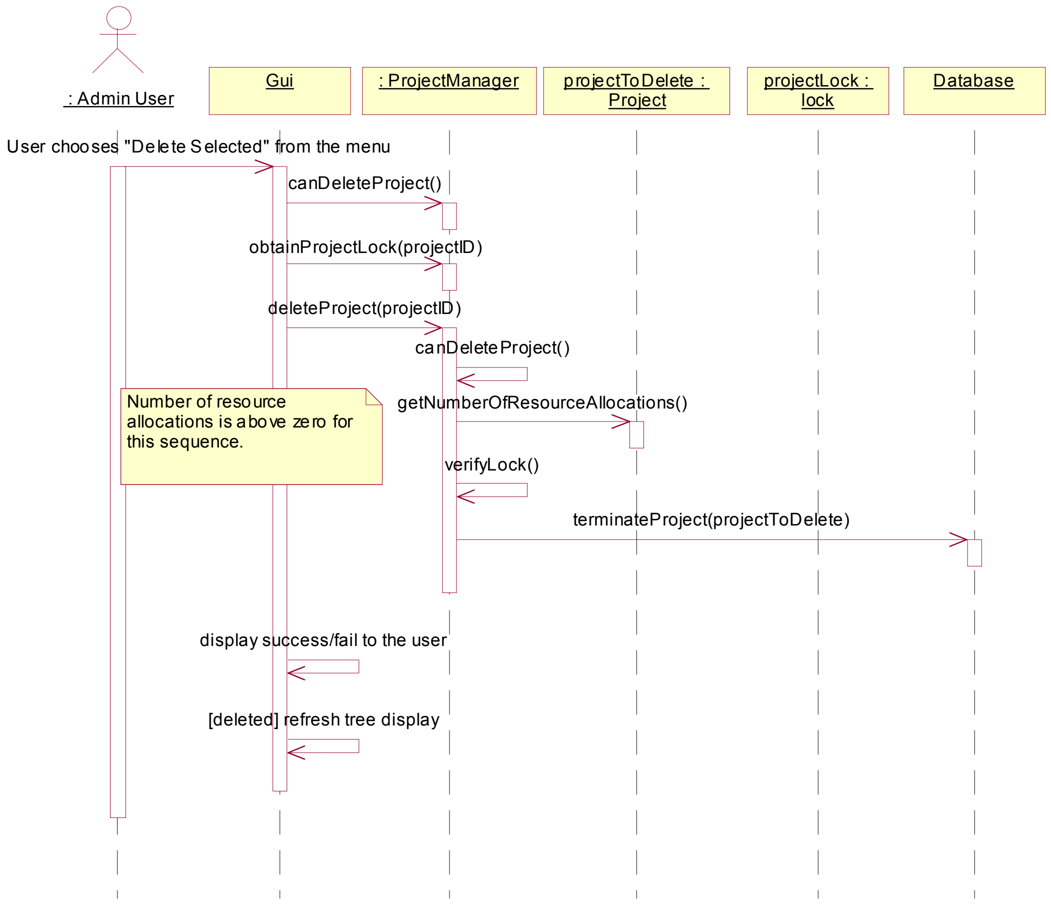
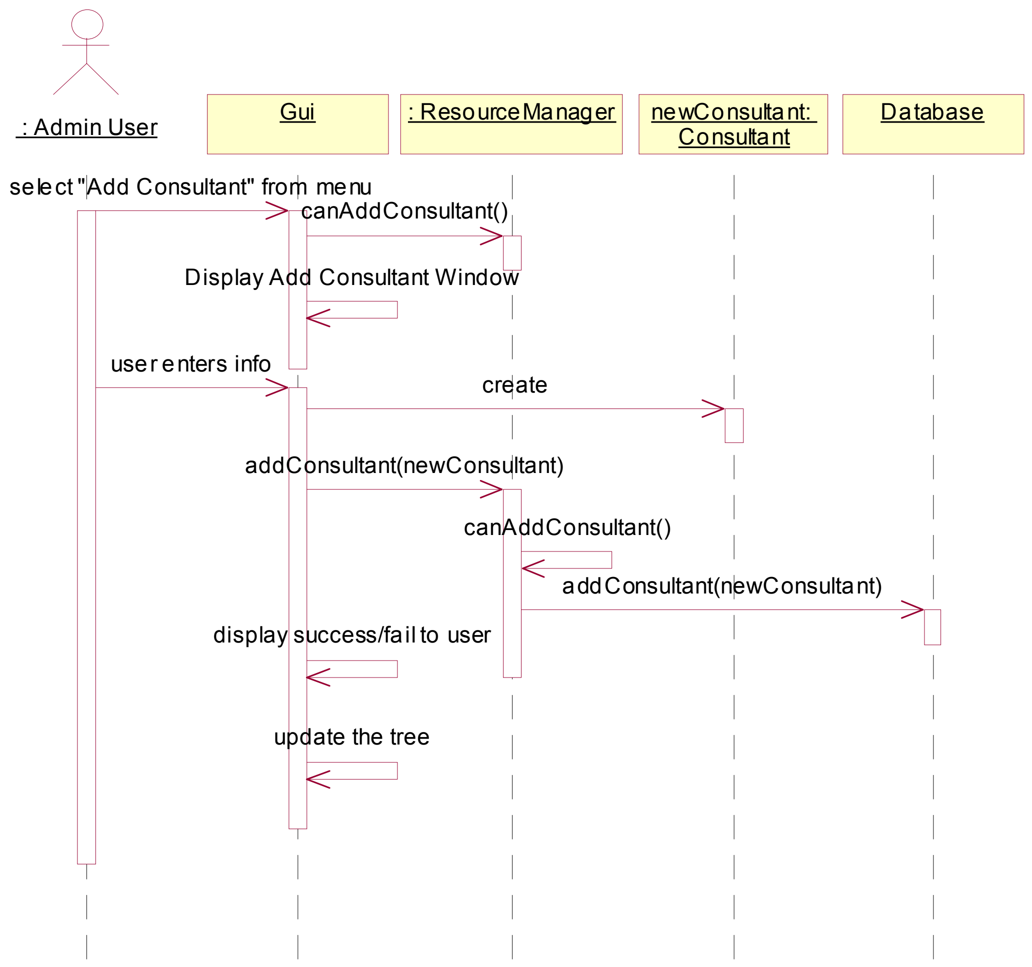

: Admin User
Gui
: ProjectManager
projectToDelete : Project
projectLock : lock
Database
User chooses "Delete Selected" from the menu
canDeleteProject()
obtainProjectLock(projectID)
deleteProject(projectID)
canDeleteProject()
Number of resource allocations is above zero for this sequence.
getNumberOfResourceAllocations()
verifyLock()
terminateProject(projectToDelete)
display success/fail to the user
[deleted] refresh tree display

Figure 13: Sequence Diagram for Add Resource Use Case (Consultant Specific)



: Admin User
Gui
: ResourceManager
newConsultant: Consultant
Database
select "Add Consultant" from menu
canAddConsultant()
Display Add Consultant Window
user enters info
create
addConsultant(newConsultant)
canAddConsultant()
addConsultant(newConsultant)
display success/fail to user
update the tree

Figure 14: Sequence Diagram for Add Resource Use Case (Machine Specific)

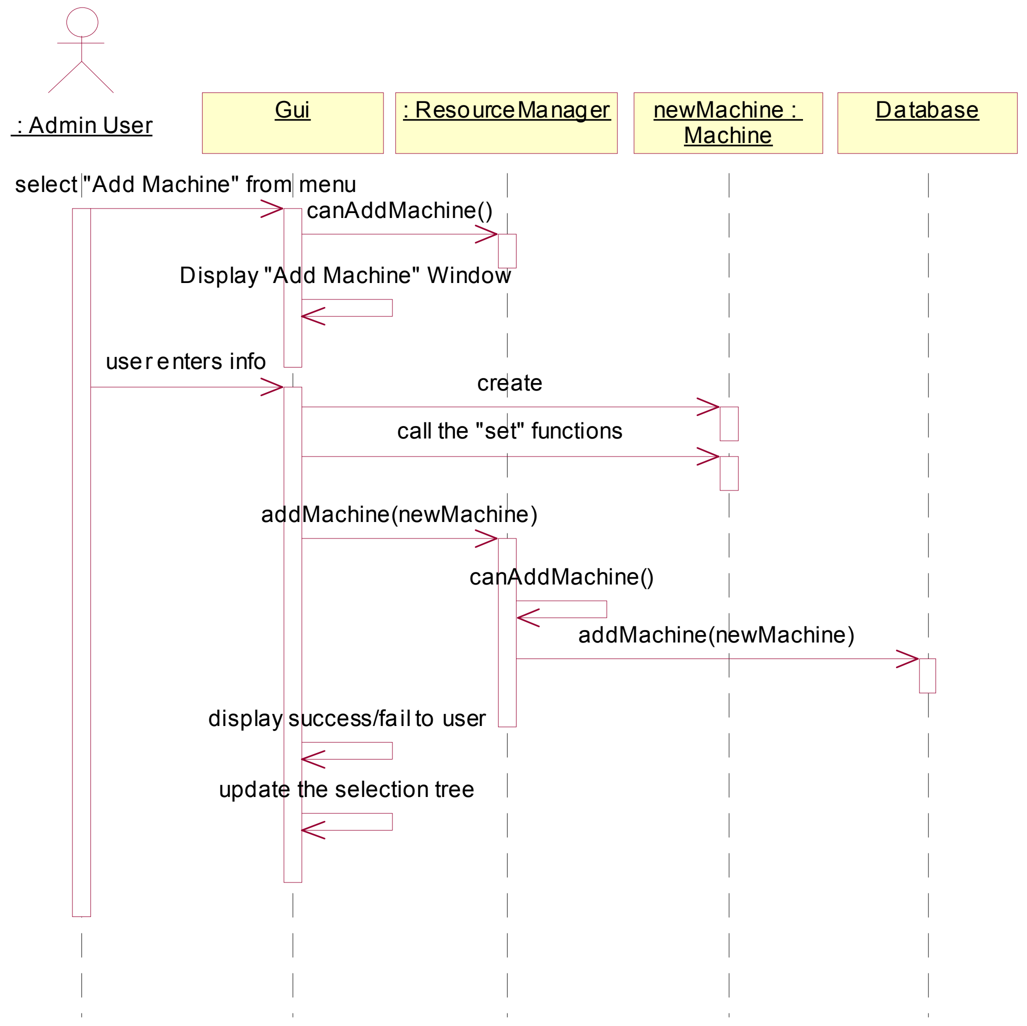


Figure 15: Sequence Diagram for Add Resource Use Case (Room Specific)

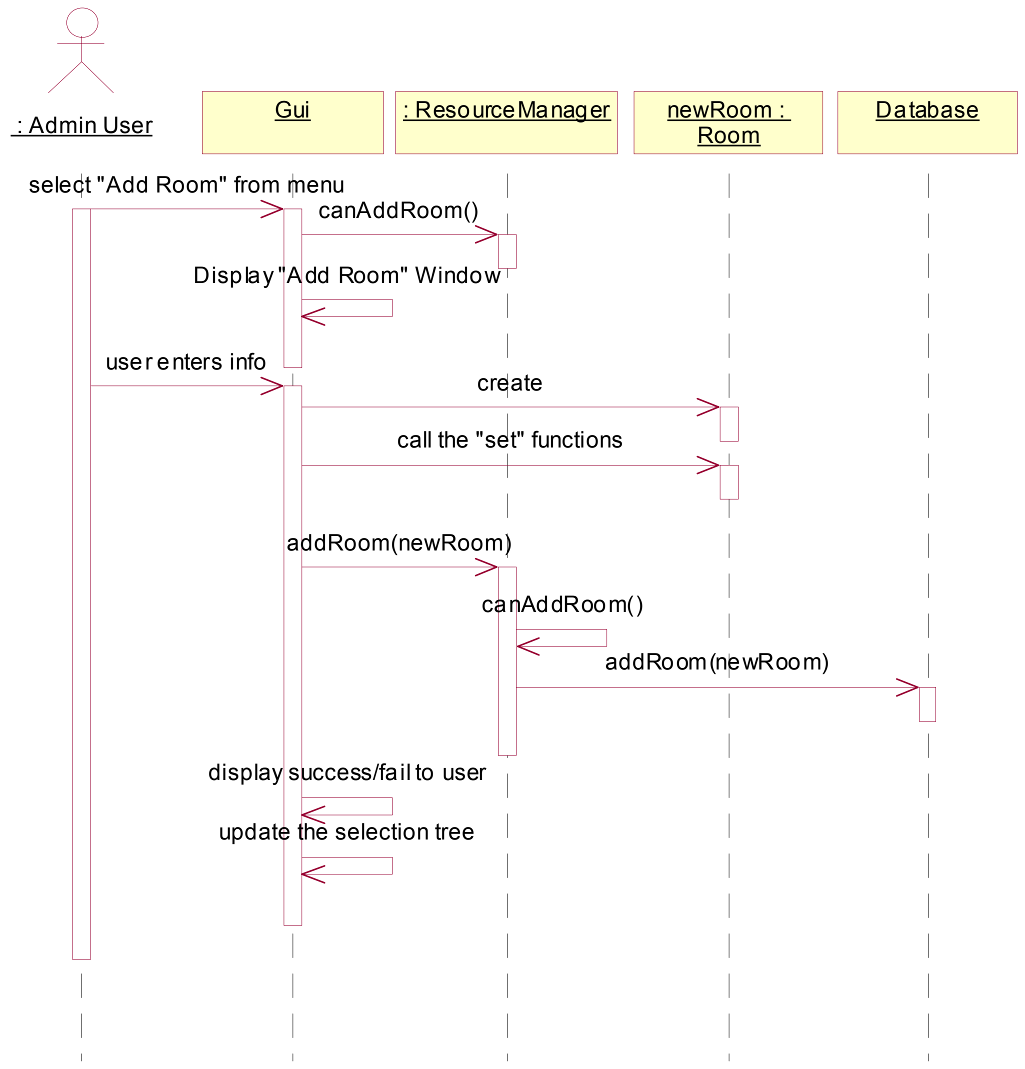


Figure 16: Sequence Diagram for Modify Resource Use Case (Consultant Specific)

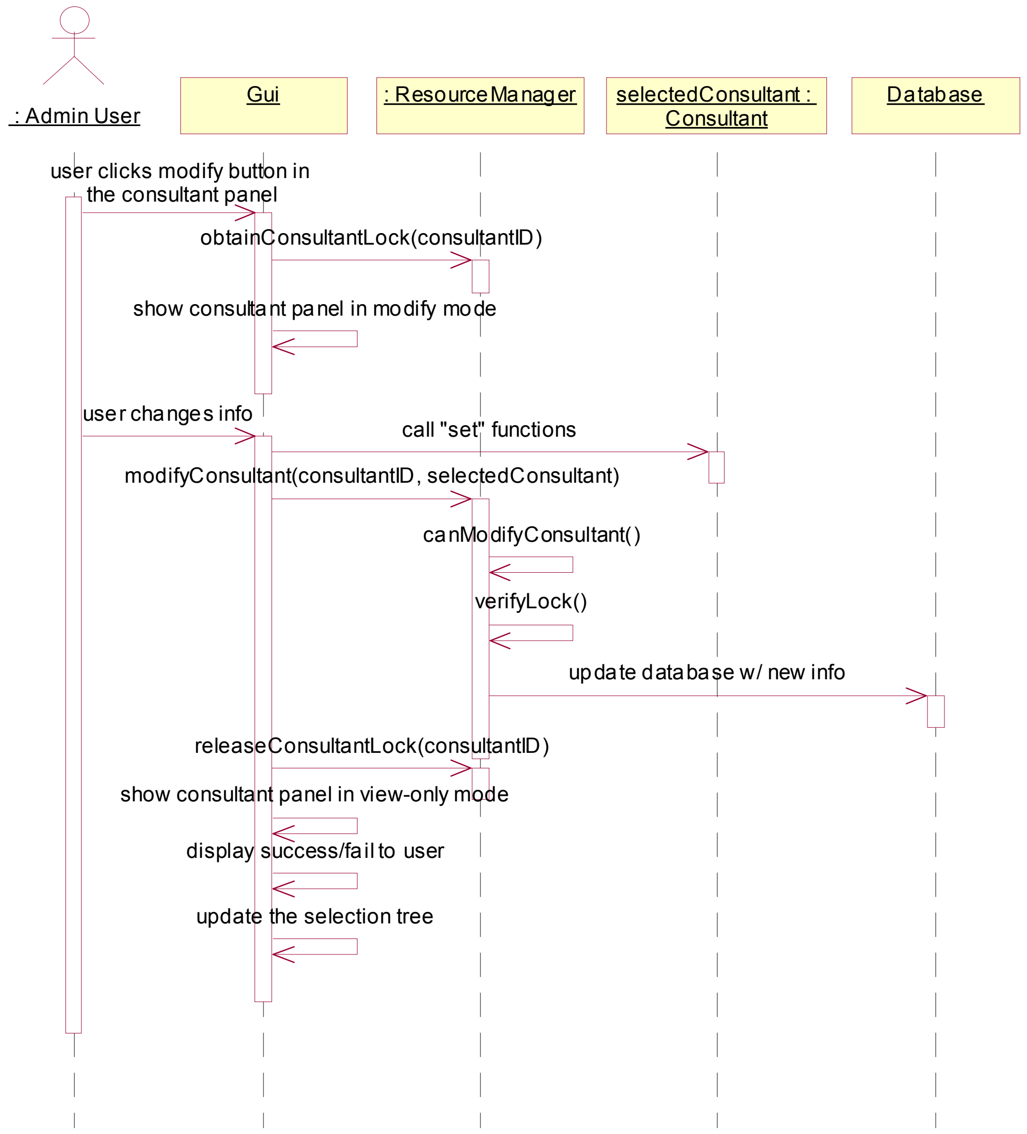

: Admin User
Gui
: ResourceManager
selectedConsultant : Consultant
Database
user clicks modify button in the consultant panel
obtainConsultantLock(consultantID)
show consultant panel in modify mode
user changes info
call "set" functions
modifyConsultant(consultantID, selectedConsultant)
canModifyConsultant()
verifyLock()
update database w/ new info
releaseConsultantLock(consultantID)
show consultant panel in view-only mode
display success/fail to user
update the selection tree

Figure 17: Sequence Diagram for Modify Resource Use Case (Machine Specific)

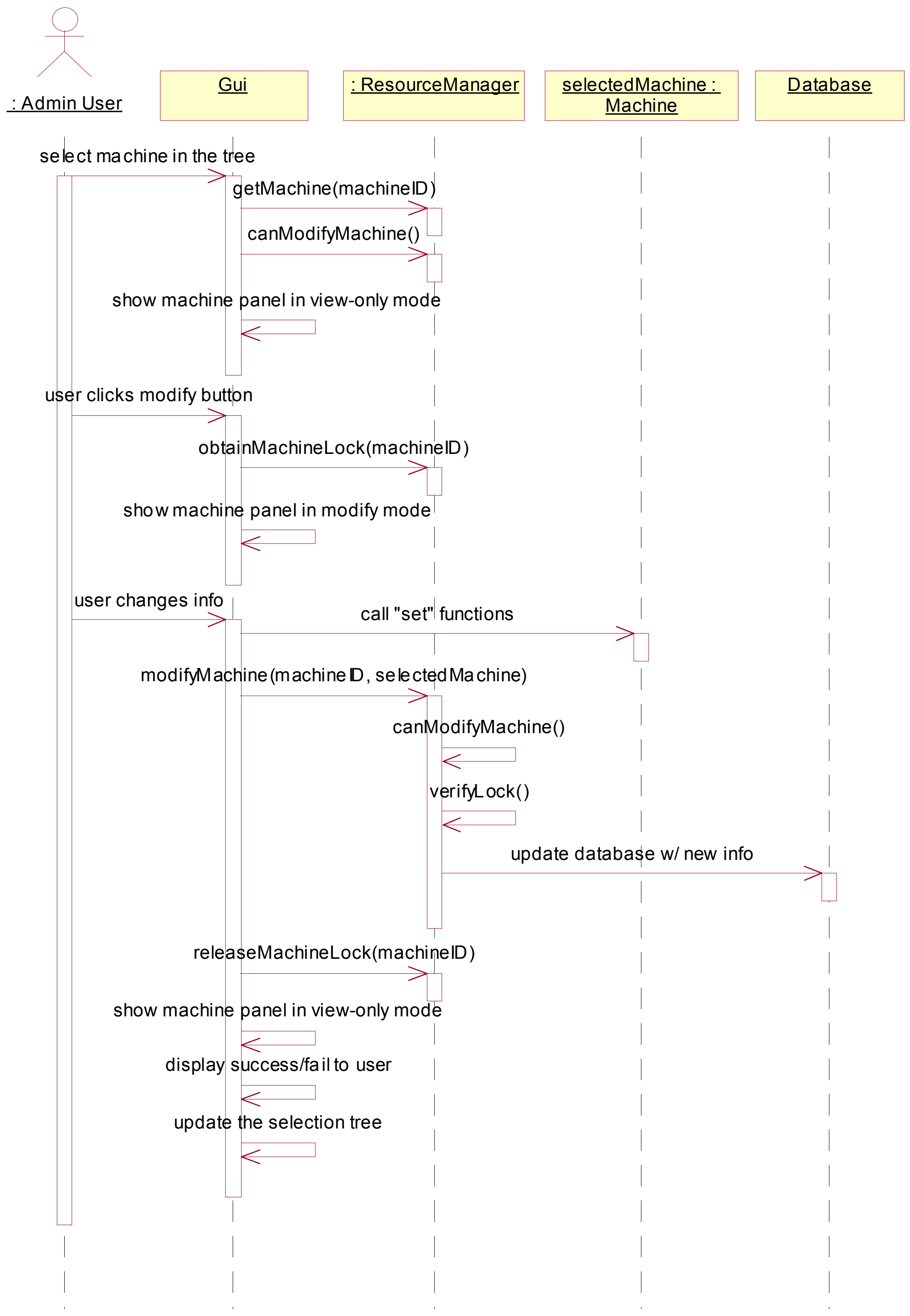

: Admin User
Gui
: ResourceManager
selectedMachine : Machine
Database
select machine in the tree
getMachine(machineID)
canModifyMachine()
show machine panel in view-only mode
user clicks modify button
obtainMachineLock(machineID)
show machine panel in modify mode
user changes info
call "set" functions
modifyMachine(machineID, selectedMachine)
canModifyMachine()
verifyLock()
update database w/ new info
releaseMachineLock(machineID)
show machine panel in view-only mode
display success/fail to user
update the selection tree

Figure 18: Sequence Diagram for Modify Resource Use Case (Room Specific)

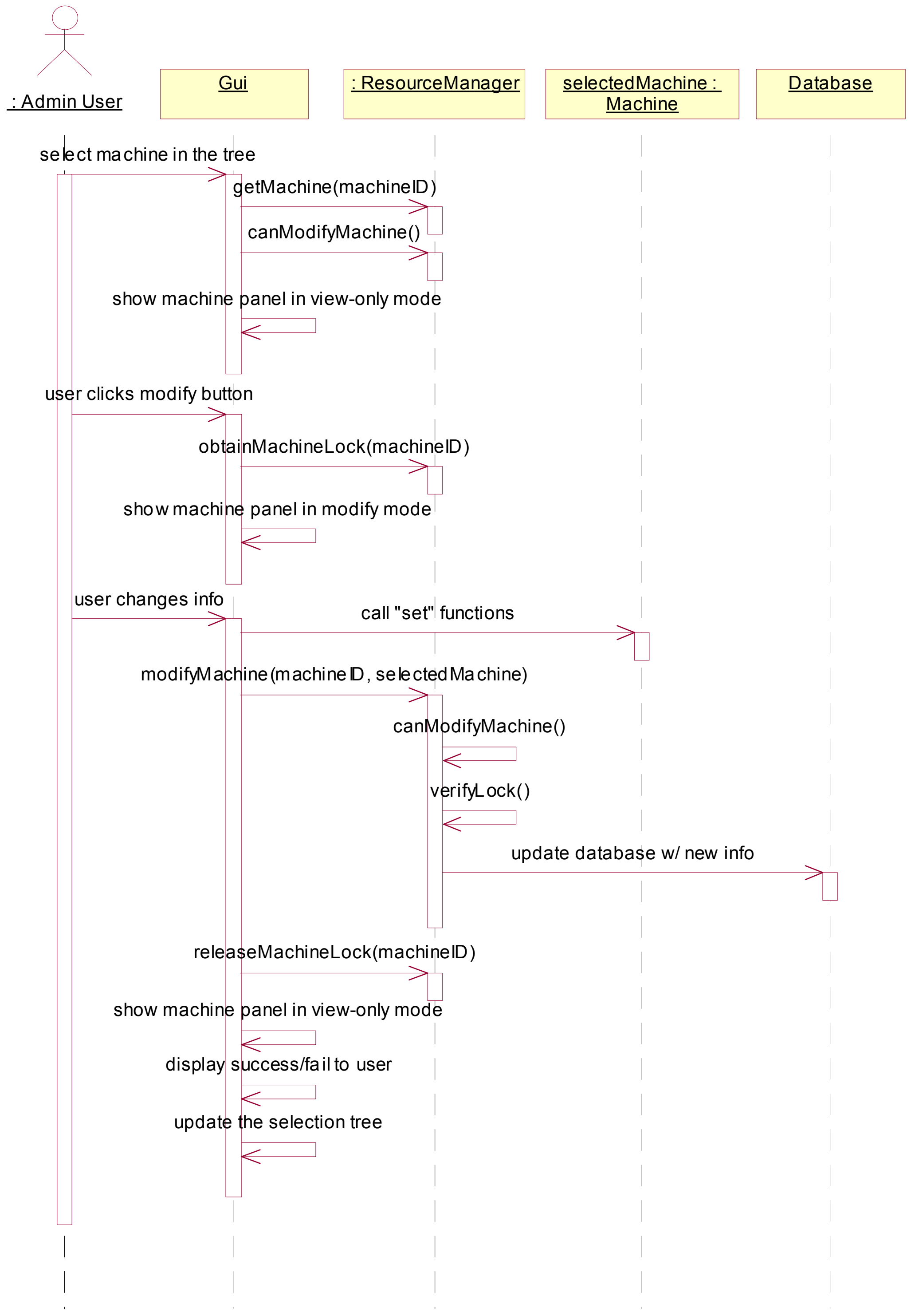

: Admin User
Gui
: ResourceManager
selectedMachine : Machine
Database
select machine in the tree
getMachine(machineID)
canModifyMachine()
show machine panel in view-only mode
user clicks modify button
obtainMachineLock(machineID)
show machine panel in modify mode
user changes info
call "set" functions
modifyMachine(machineID, selectedMachine)
canModifyMachine()
verifyLock()
update database w/ new info
releaseMachineLock(machineID)
show machine panel in view-only mode
display success/fail to user
update the selection tree

Figure 19: Sequence Diagram for Delete Resource Use Case (Consultant Specific)

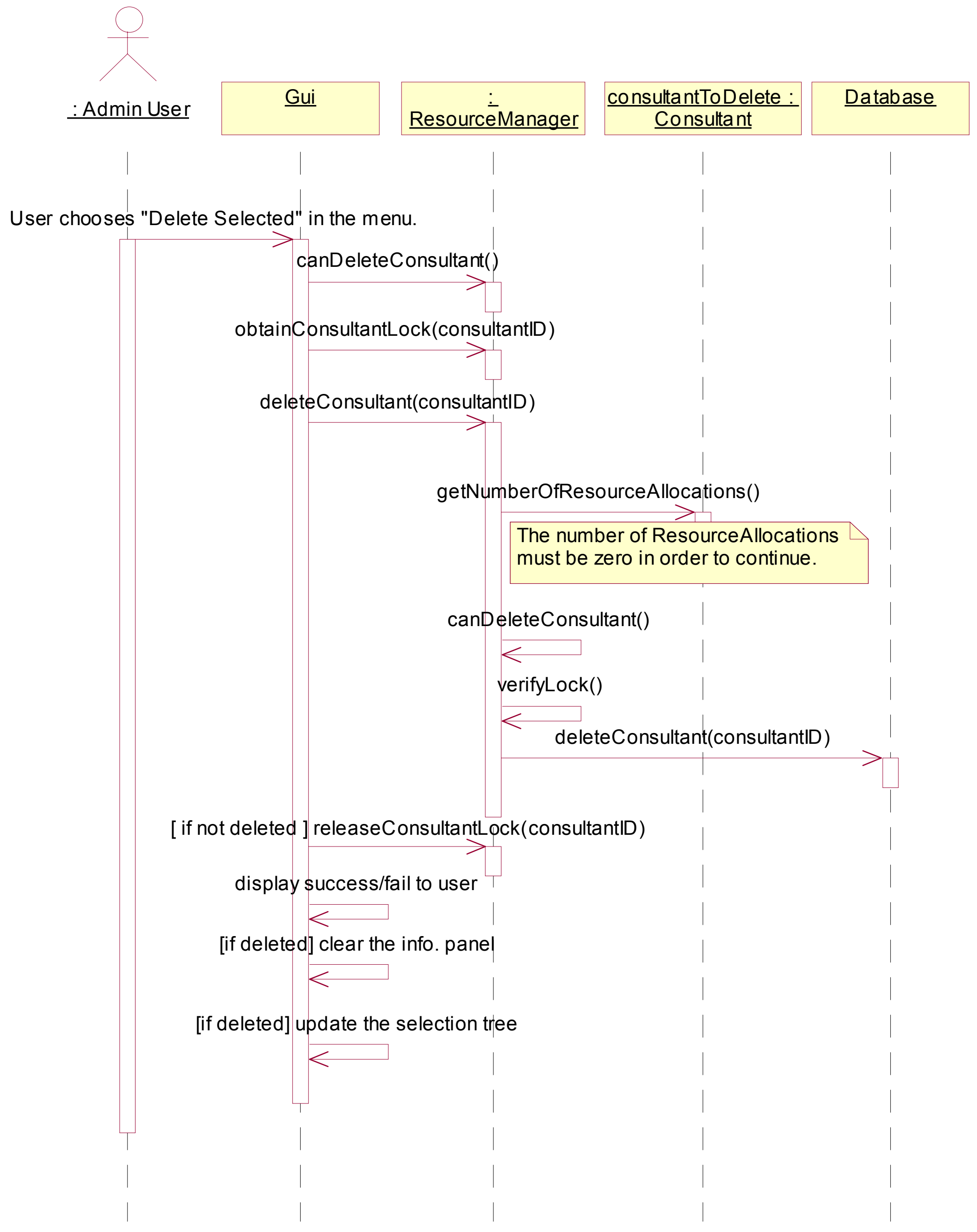

: Admin User
Gui
: ResourceManager
consultantToDelete : Consultant
Database
User chooses "Delete Selected" in the menu.
canDeleteConsultant()
obtainConsultantLock(consultantID)
deleteConsultant(consultantID)
getNumberOfResourceAllocations()
The number of ResourceAllocations must be zero in order to continue.
canDeleteConsultant()
verifyLock()
deleteConsultant(consultantID)
[if not deleted] releaseConsultantLock(consultantID)
display success/fail to user
[if deleted] clear the info. panel
[if deleted] update the selection tree

Figure 20: Sequence Diagram for Delete Resource Use Case (Machine Specific)

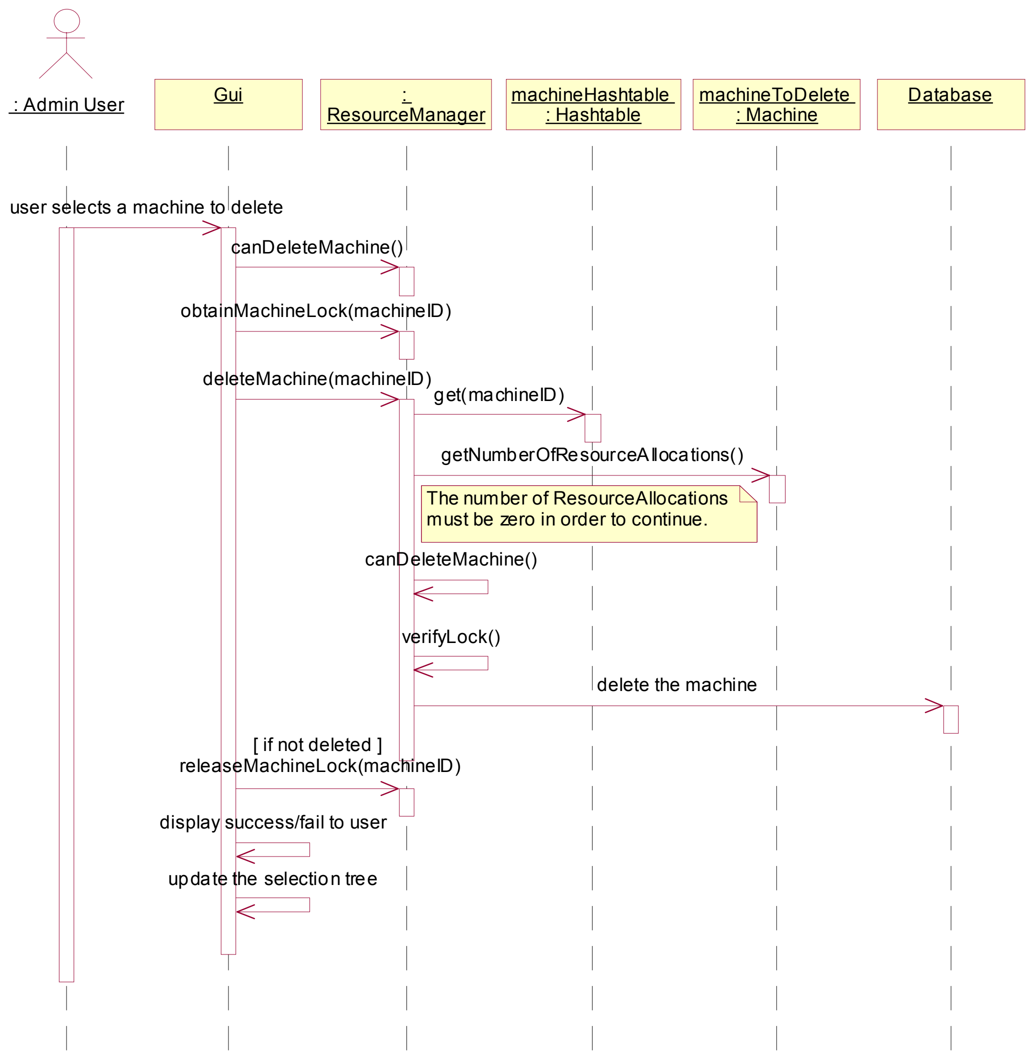

: Admin User
Gui
: ResourceManager
machineHashtable : Hashtable
machineToDelete : Machine
Database
user selects a machine to delete
canDeleteMachine()
obtainMachineLock(machineID)
deleteMachine(machineID)
get(machineID)
getNumberOfResourceAllocations()
The number of ResourceAllocations must be zero in order to continue.
canDeleteMachine()
verifyLock()
delete the machine
[if not deleted] releaseMachineLock(machineID)
display success/fail to user
update the selection tree

Figure 21: Sequence Diagram for Delete Resource Use Case (Room Specific)

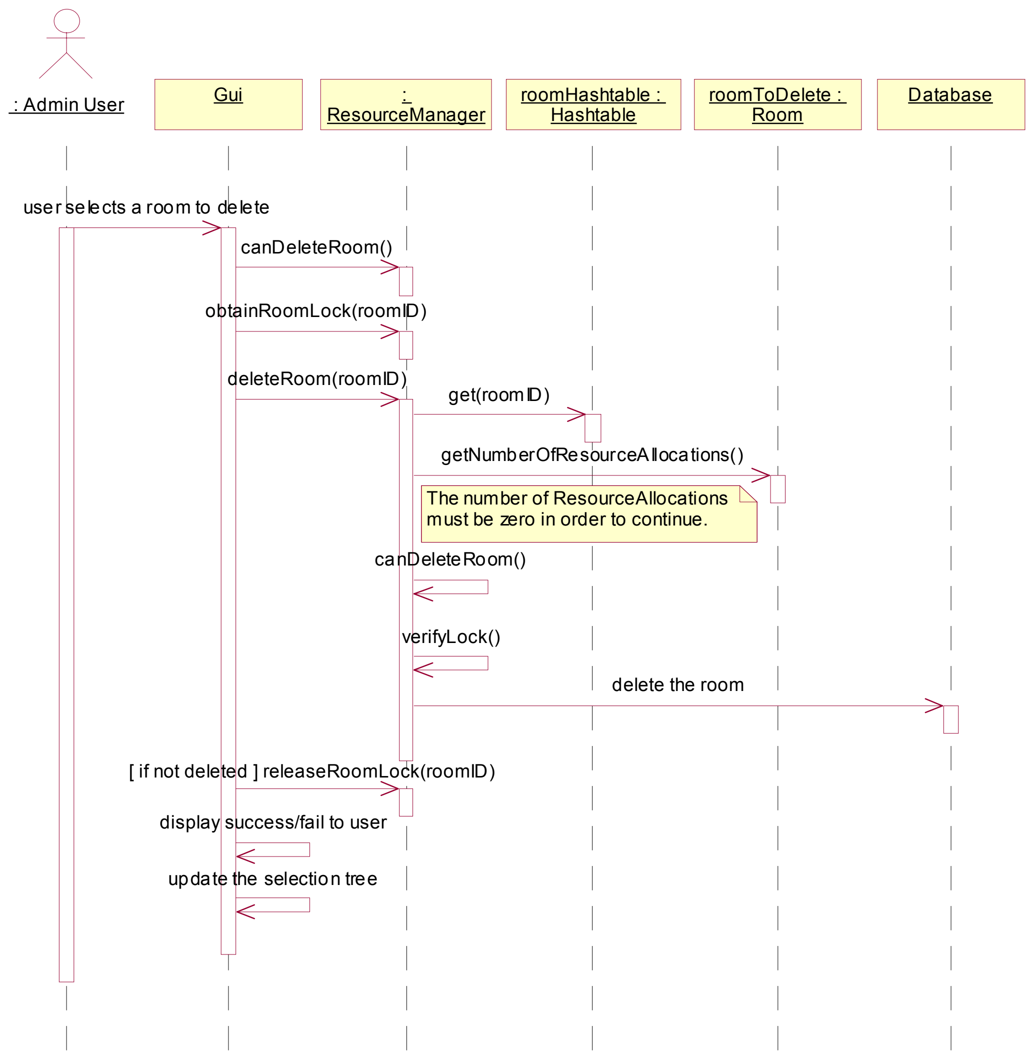

: Admin User
Gui
: ResourceManager
roomHashtable : Hashtable
roomToDelete : Room
Database
user selects a room to delete
canDeleteRoom()
obtainRoomLock(roomID)
deleteRoom(roomID)
get(roomID)
getNumberOfResourceAllocations()
The number of ResourceAllocations must be zero in order to continue.
canDeleteRoom()
verifyLock()
delete the room
[if not deleted] releaseRoomLock(roomID)
display success/fail to user
update the selection tree

Figure 22: Sequence Diagram for Add Resource Allocation Use Case

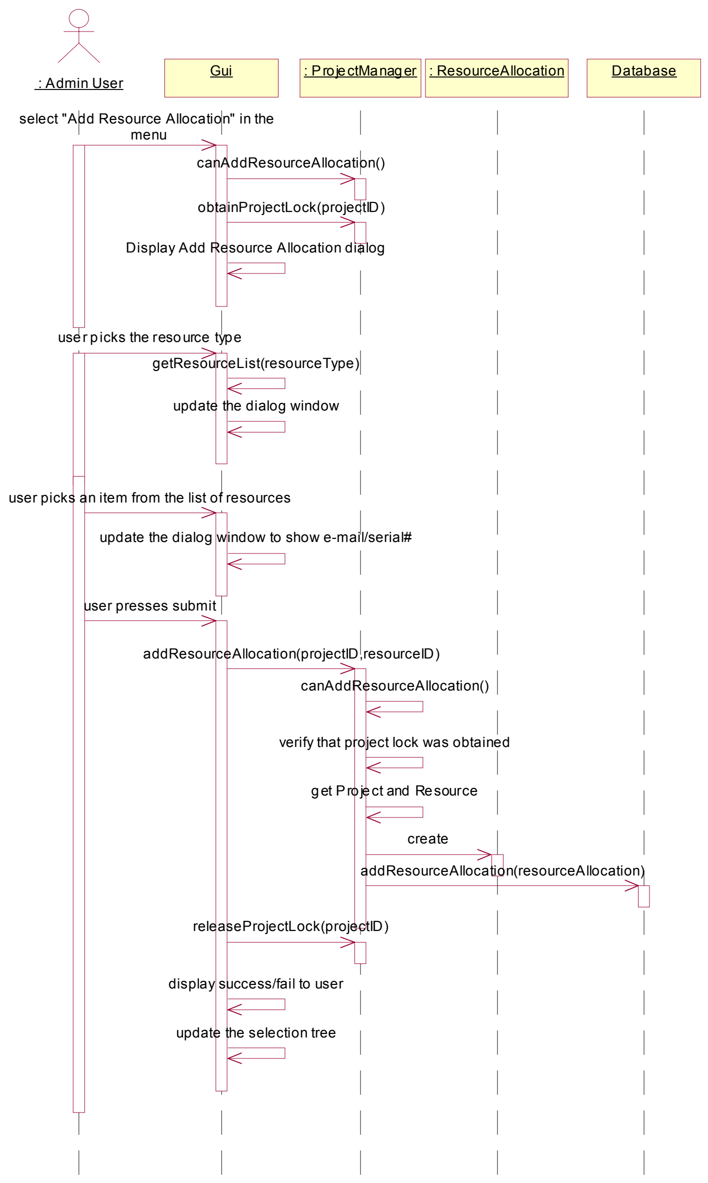

: Admin User
Gui
: ProjectManager
: ResourceAllocation
Database
select "Add Resource Allocation" in the menu
canAddResourceAllocation()
obtainProjectLock(projectID)
Display Add Resource Allocation dialog
user picks the resource type
getResourceList(resourceType)
update the dialog window
user picks an item from the list of resources
update the dialog window to show e-mail/serial#
user presses submit
addResourceAllocation(projectID,resourceID)
canAddResourceAllocation()
verify that project lock was obtained
get Project and Resource
create
addResourceAllocation(resourceAllocation)
releaseProjectLock(projectID)
display success/fail to user
update the selection tree

Figure 23: Sequence Diagram for Add Resource Allocation Use Case (Low-Level)

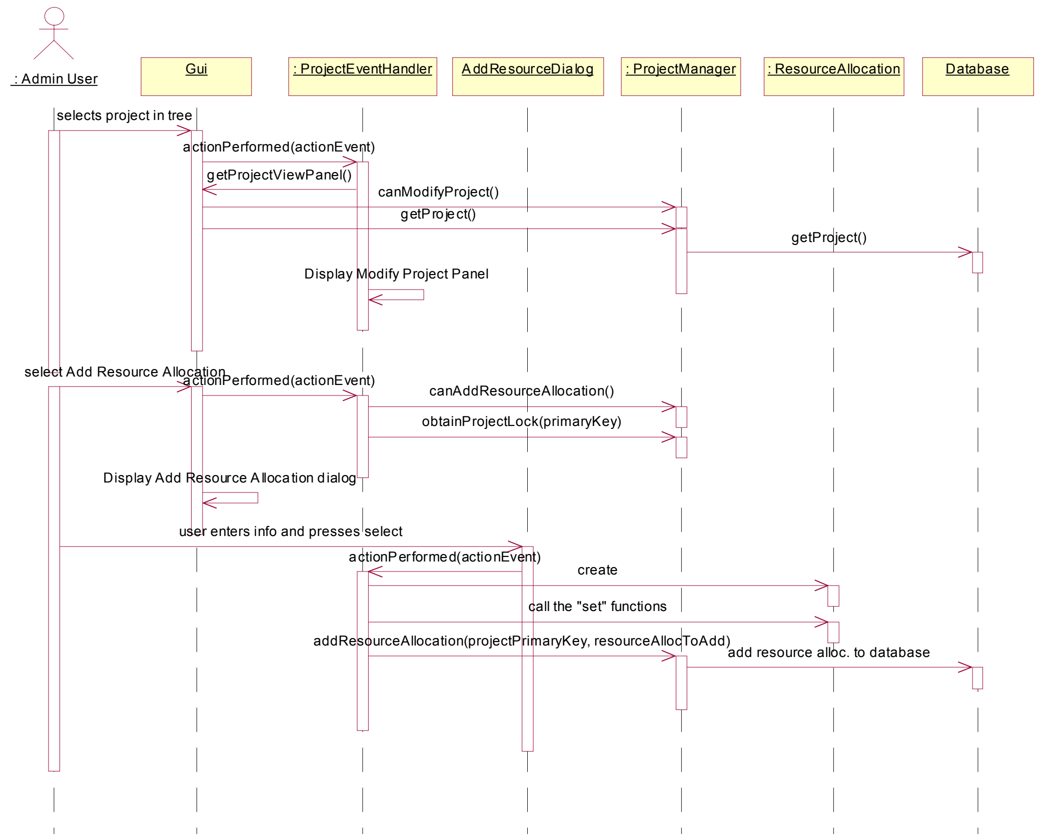

: Admin User
Gui
: ProjectEventHandler
AddResourceDialog
: ProjectManager
: ResourceAllocation
Database
selects project in tree
actionPerformed(actionEvent)
getProjectViewPanel()
canModifyProject()
getProject()
getProject()
Display Modify Project Panel
select Add Resource Allocation
actionPerformed(actionEvent)
canAddResourceAllocation()
obtainProjectLock(primaryKey)
Display Add Resource Allocation dialog
user enters info and presses select
actionPerformed(actionEvent)
create
call the "set" functions
addResourceAllocation(projectPrimaryKey, resourceAllocToAdd)
add resource alloc. to database

Figure 24: Sequence Diagram for Delete Resource Allocation Use Case

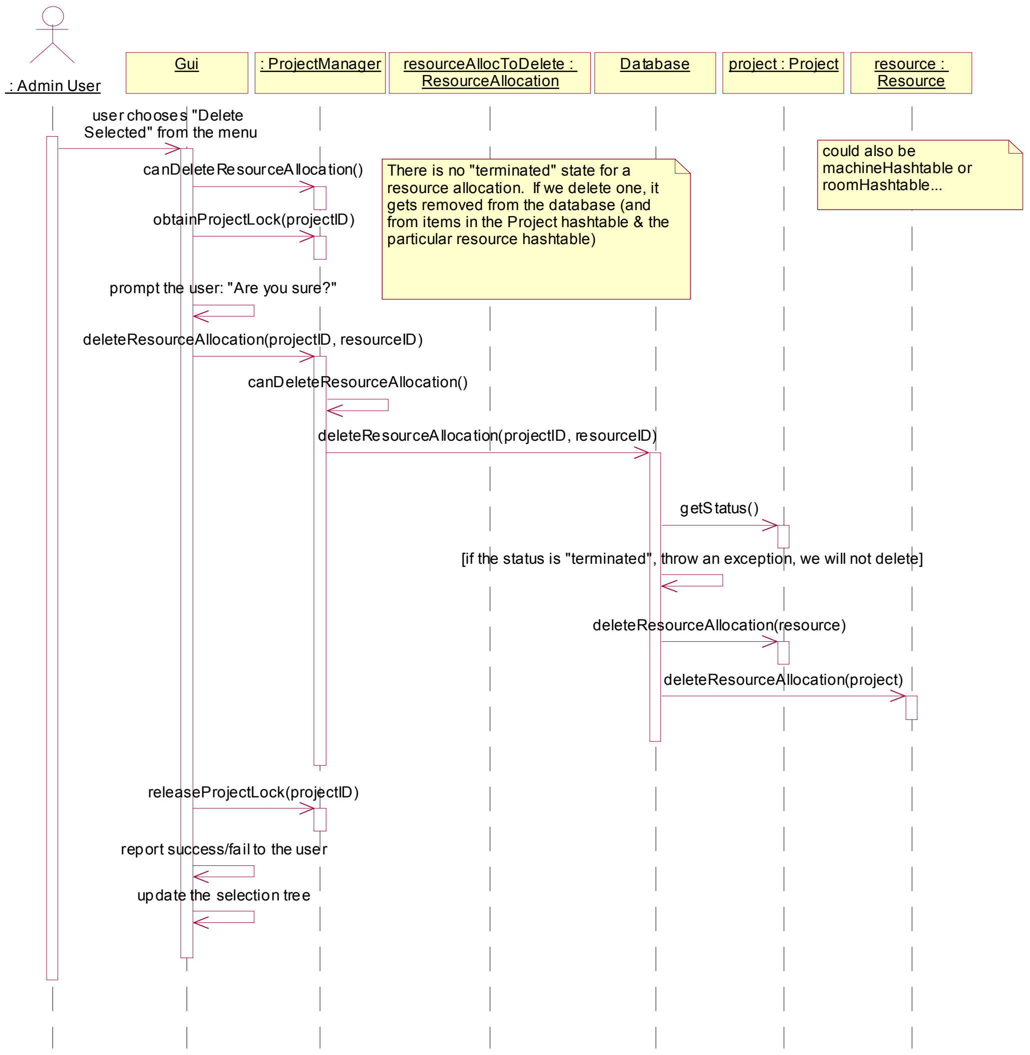

: Admin User
Gui
: ProjectManager
resourceAllocToDelete : ResourceAllocation
Database
project : Project
resource : Resource
user chooses "Delete Selected" from the menu
canDeleteResourceAllocation()
obtainProjectLock(projectID)
There is no "terminated" state for a resource allocation. If we delete one, it gets removed from the database (and from items in the Project hashtable & the particular resource hashtable)
could also be machineHashtable or roomHashtable...
prompt the user: "Are you sure?"
deleteResourceAllocation(projectID, resourceID)
canDeleteResourceAllocation()
deleteResourceAllocation(projectID, resourceID)
getStatus()
[if the status is "terminated", throw an exception, we will not delete]
deleteResourceAllocation(resource)
deleteResourceAllocation(project)
releaseProjectLock(projectID)
report success/fail to the user
update the selection tree

Figure 25: Sequence Diagram for Add Timeslot Use Case

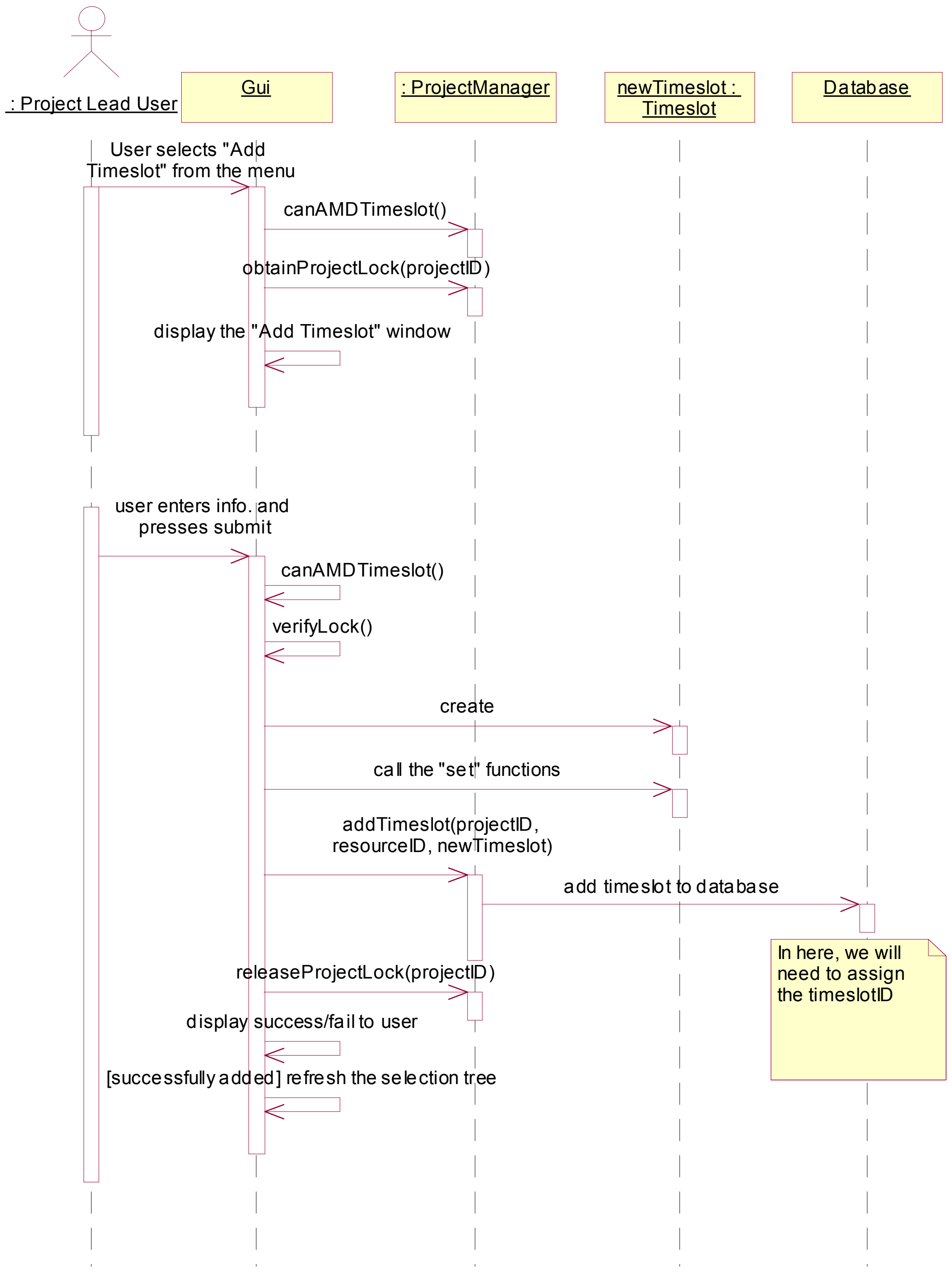

: Project Lead User
Gui
: ProjectManager
newTimeslot : Timeslot
Database
User selects "Add Timeslot" from the menu
canAMDTimeslot()
obtainProjectLock(projectID)
display the "Add Timeslot" window
user enters info. and presses submit
canAMDTimeslot()
verifyLock()
create
call the "set" functions
addTimeslot(projectID, resourceID, newTimeslot)
add timeslot to database
In here, we will need to assign the timeslotID
releaseProjectLock(projectID)
display success/fail to user
[successfully added] refresh the selection tree

Figure 26: Sequence Diagram for Modify Timeslot Use Case

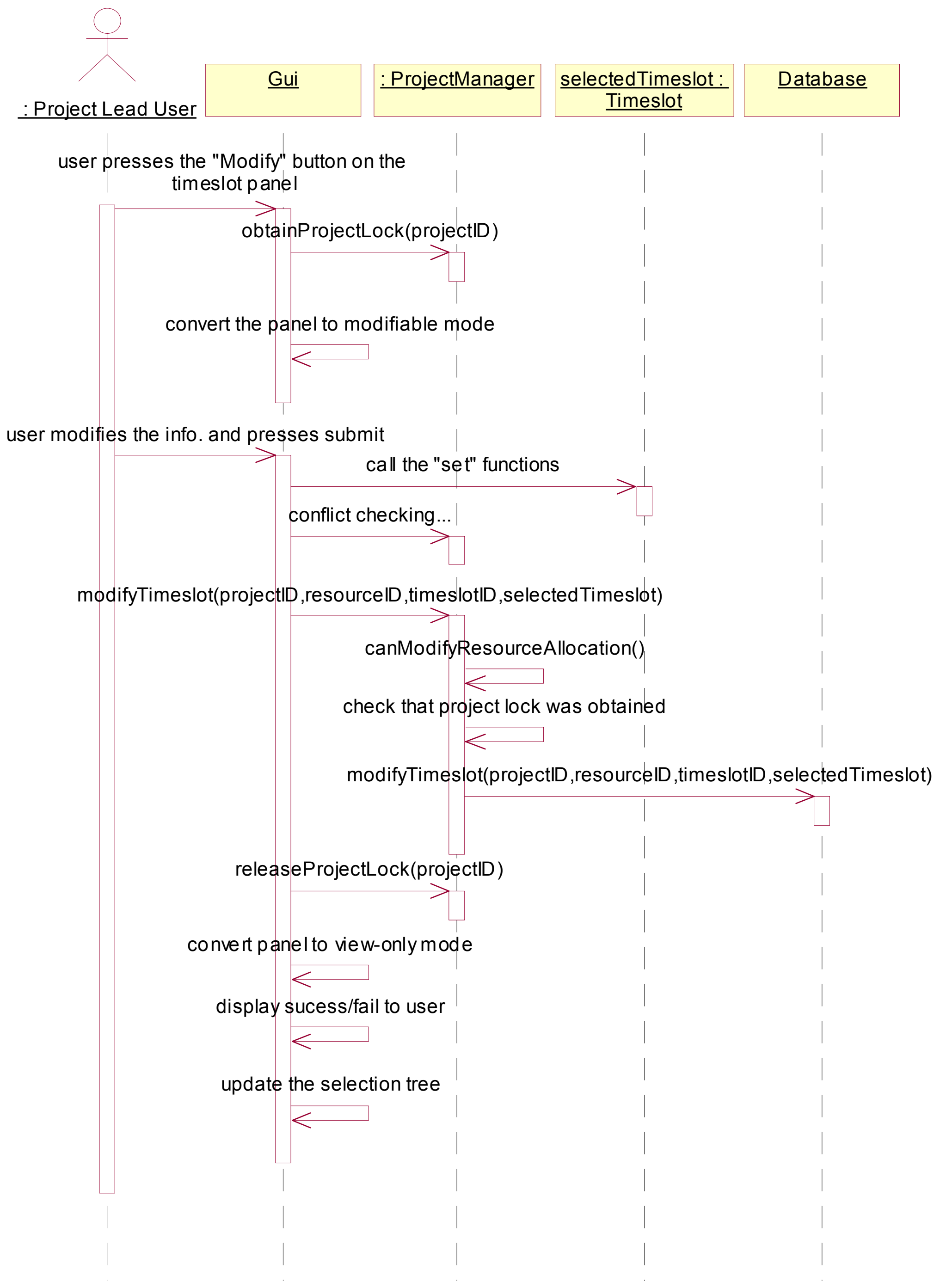

: Project Lead User
Gui
: ProjectManager
selectedTimeslot : Timeslot
Database
user presses the "Modify" button on the timeslot panel
obtainProjectLock(projectID)
convert the panel to modifiable mode
user modifies the info. and presses submit
call the "set" functions
conflict checking...
modifyTimeslot(projectID,resourceID,timeslotID,selectedTimeslot)
canModifyResourceAllocation()
check that project lock was obtained
modifyTimeslot(projectID,resourceID,timeslotID,selectedTimeslot)
releaseProjectLock(projectID)
convert panel to view-only mode
display sucess/fail to user
update the selection tree

Figure 27: Sequence Diagram for Delete Timeslot Use Case

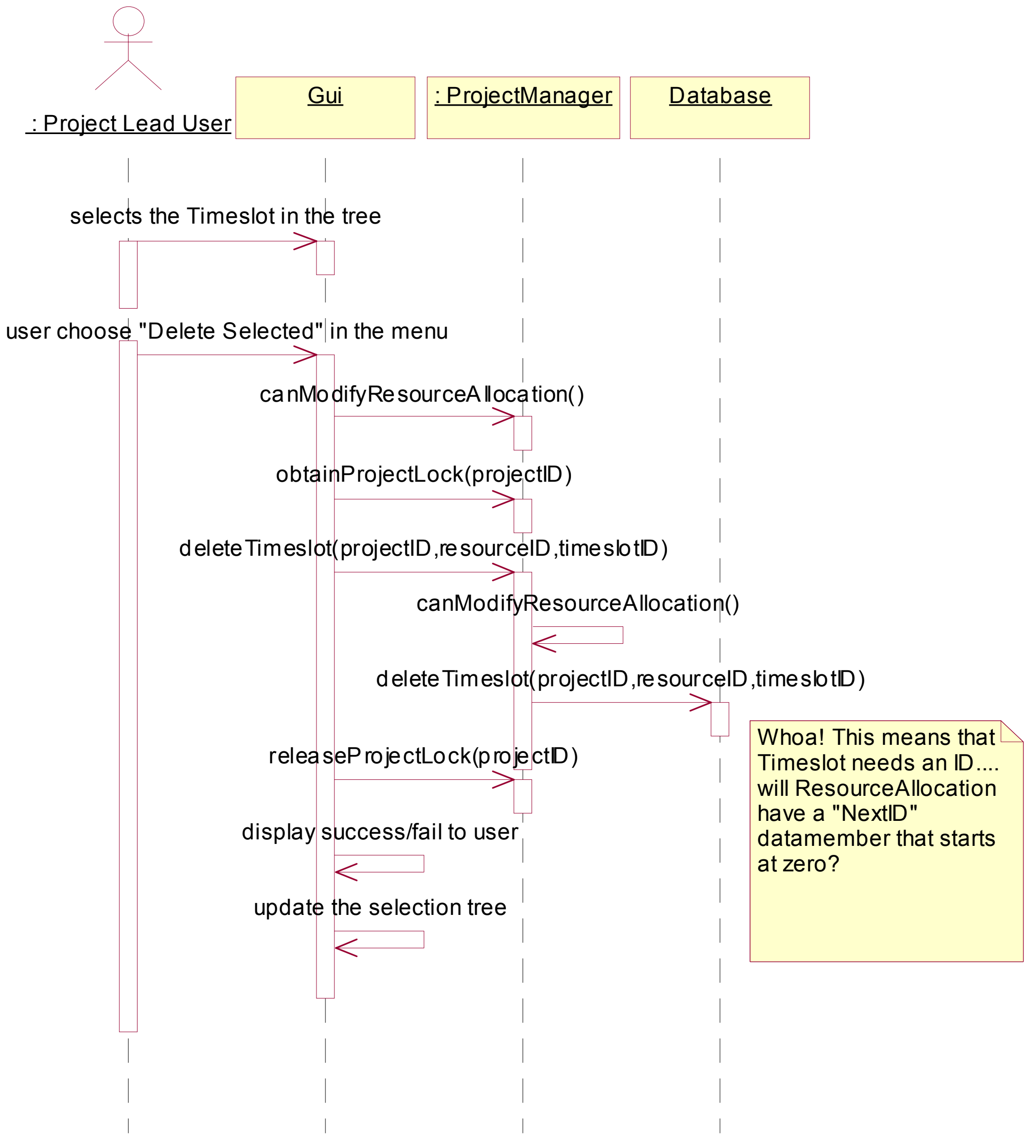

: Project Lead User
Gui
: ProjectManager
Database
selects the Timeslot in the tree
user choose "Delete Selected" in the menu
canModifyResourceAllocation()
obtainProjectLock(projectID)
deleteTimeslot(projectID,resourceID,timeslotID)
canModifyResourceAllocation()
deleteTimeslot(projectID,resourceID,timeslotID)
releaseProjectLock(projectID)
display success/fail to user
update the selection tree
Whoa! This means that Timeslot needs an ID.... will ResourceAllocation have a "NextID" datamember that starts at zero?

Figure 28: Sequence Diagram for Add User Use Case

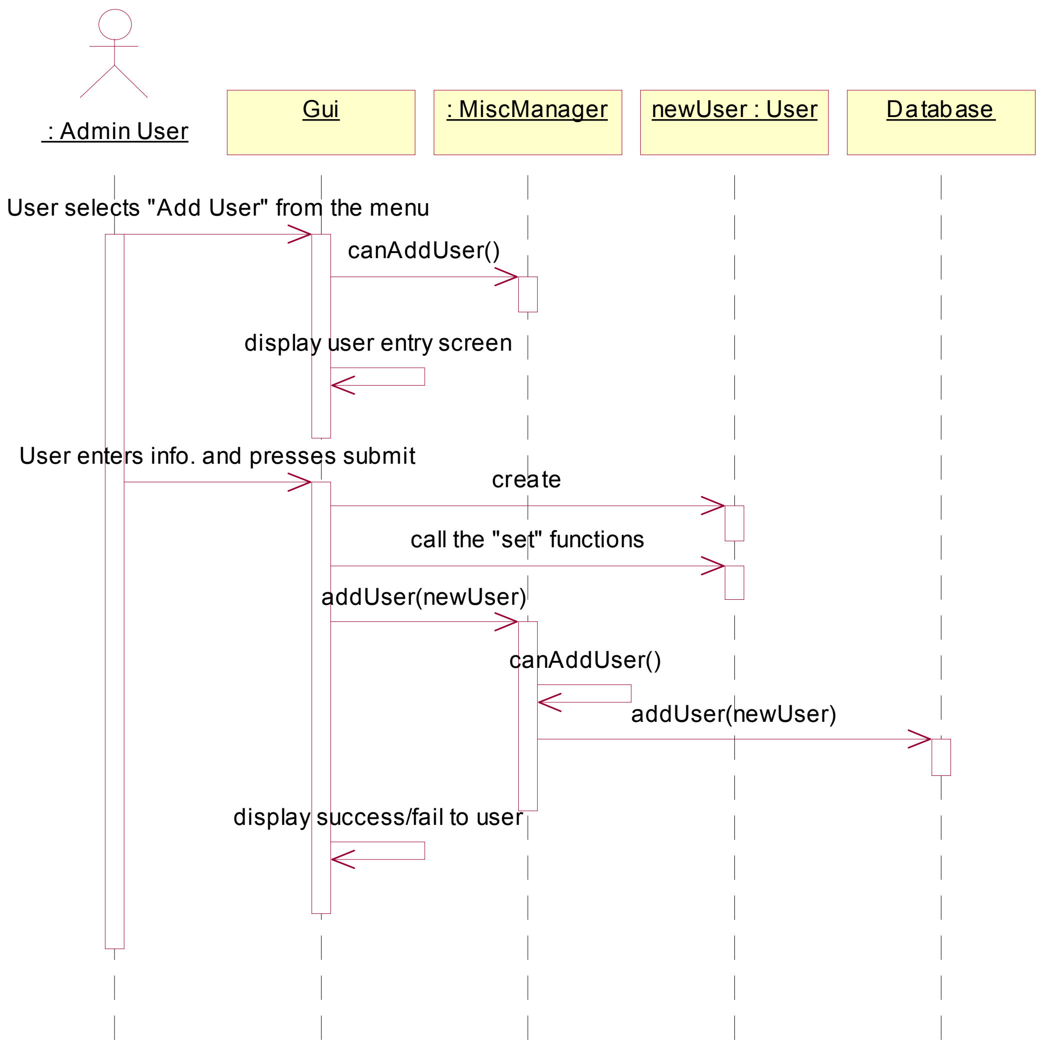


Figure 29: Sequence Diagram for Modify User Use Case

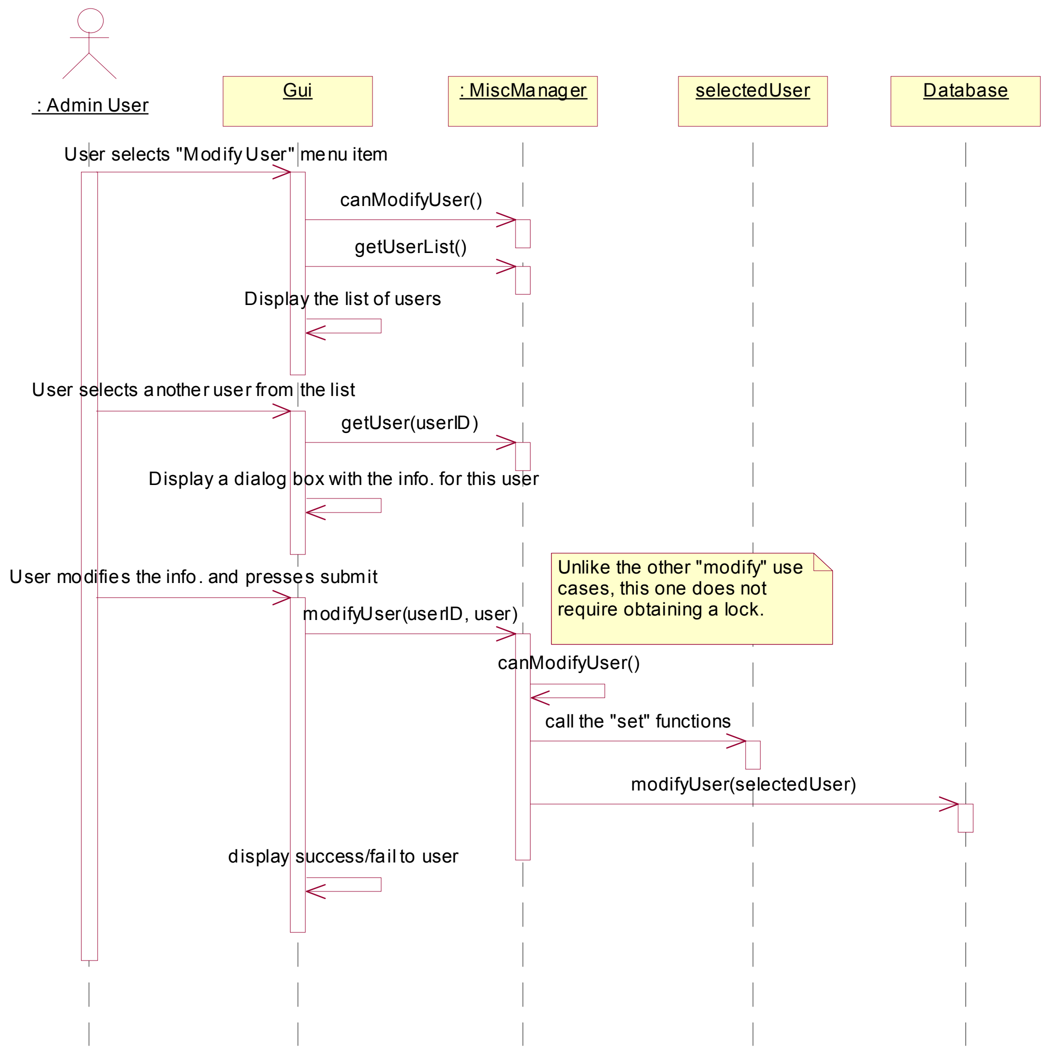

: Admin User
Gui
: MiscManager
selectedUser
Database
User selects "Modify User" menu item
canModifyUser()
getUserList()
Display the list of users
User selects another user from the list
getUser(userID)
Display a dialog box with the info. for this user
User modifies the info. and presses submit
modifyUser(userID, user)
Unlike the other "modify" use cases, this one does not require obtaining a lock.
canModifyUser()
call the "set" functions
modifyUser(selectedUser)
display success/fail to user

Figure 30: Sequence Diagram for Delete User Use Case

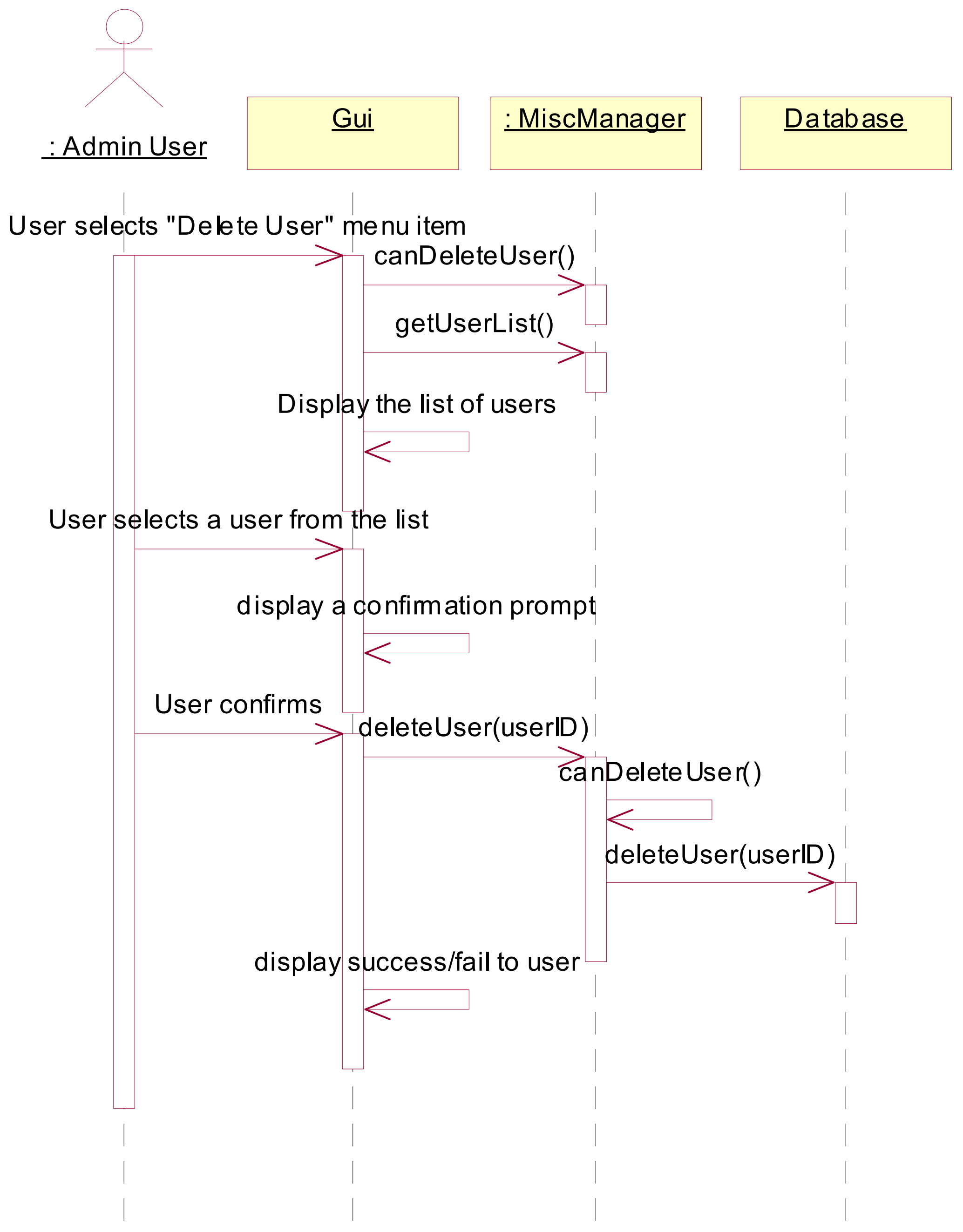

: Admin User
Gui
: MiscManager
Database
User selects "Delete User" menu item
canDeleteUser()
getUserList()
Display the list of users
User selects a user from the list
display a confirmation prompt
User confirms
deleteUser(userID)
canDeleteUser()
deleteUser(userID)
display success/fail to user

Figure 31: Sequence Diagram for Change Password Use Case

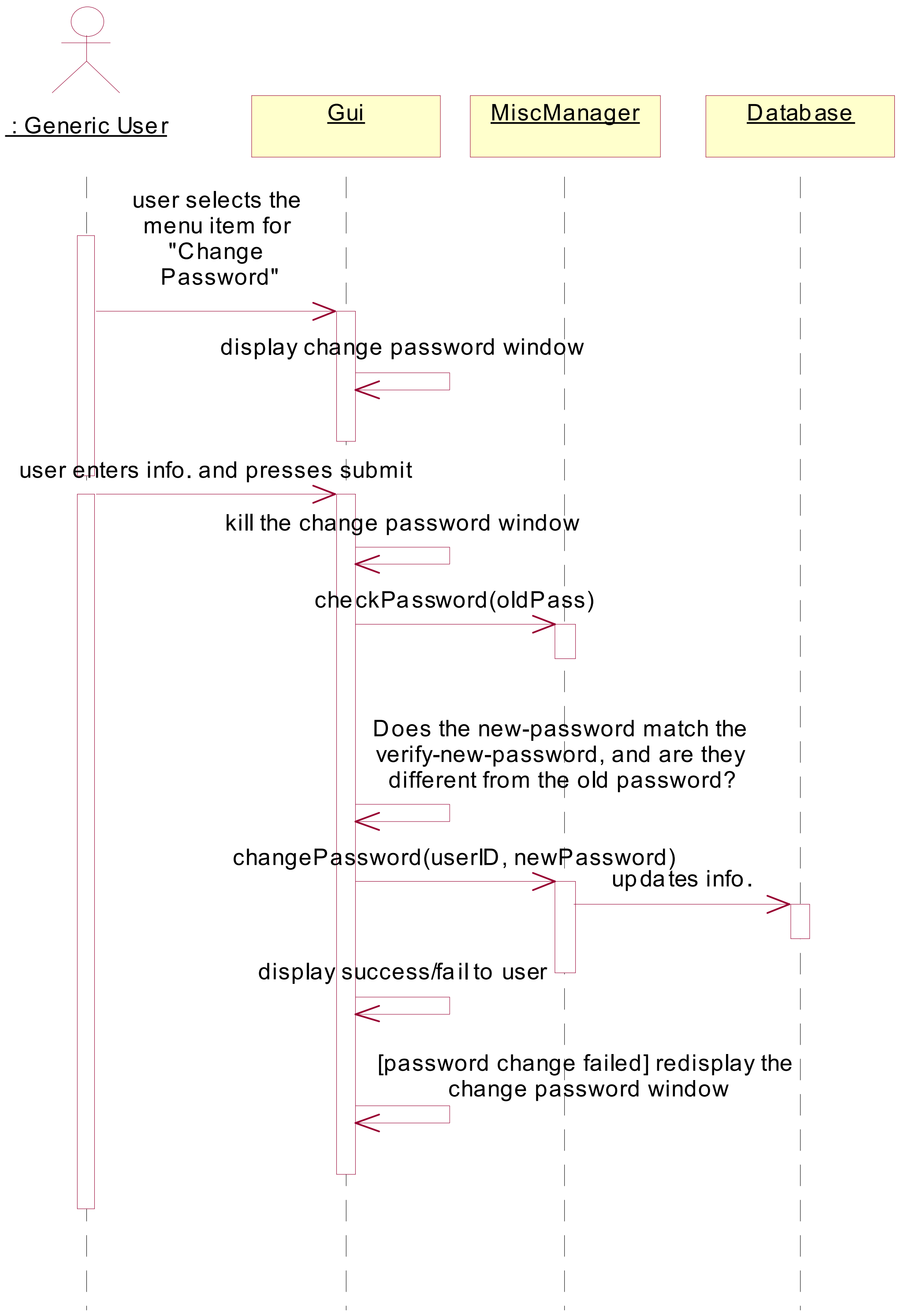

: Generic User
Gui
MiscManager
Database
user selects the menu item for "Change Password"
display change password window
user enters info. and presses submit
kill the change password window
checkPassword(oldPass)
Does the new-password match the verify-new-password, and are they different from the old password?
changePassword(userID, newPassword)
updates info.
display success/fail to user
[password change failed] redisplay the change password window

Figure 32: Sequence Diagram for Startup GUI Use Case

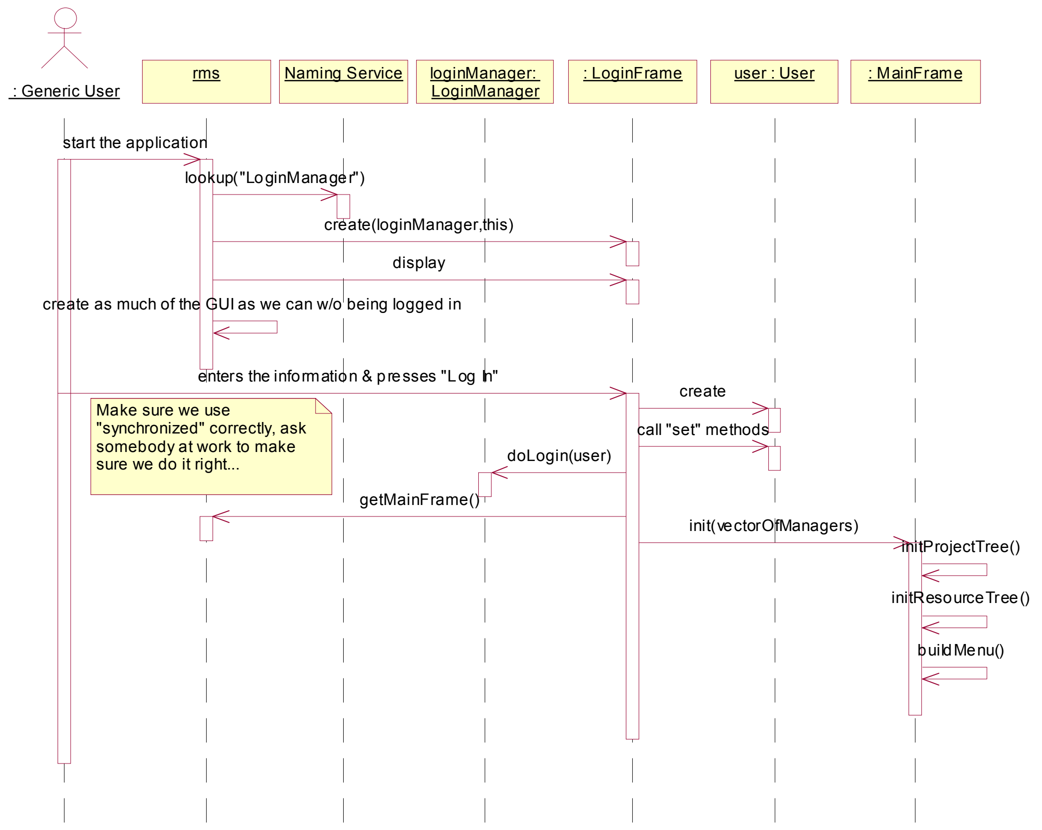

: Generic User
rms
Naming Service
loginManager: LoginManager
: LoginFrame
user : User
: MainFrame
start the application
lookup("LoginManager")
create(loginManager,this)
display
create as much of the GUI as we can w/o being logged in
enters the information & presses "Log In"
create
call "set" methods
doLogin(user)
getMainFrame()
init(vectorOfManagers)
initProjectTree()
initResourceTree()
buildMenu()
Make sure we use "synchronized" correctly, ask somebody at work to make sure we do it right...

Figure 33: Sequence Diagram for Shutdown GUI Use Case

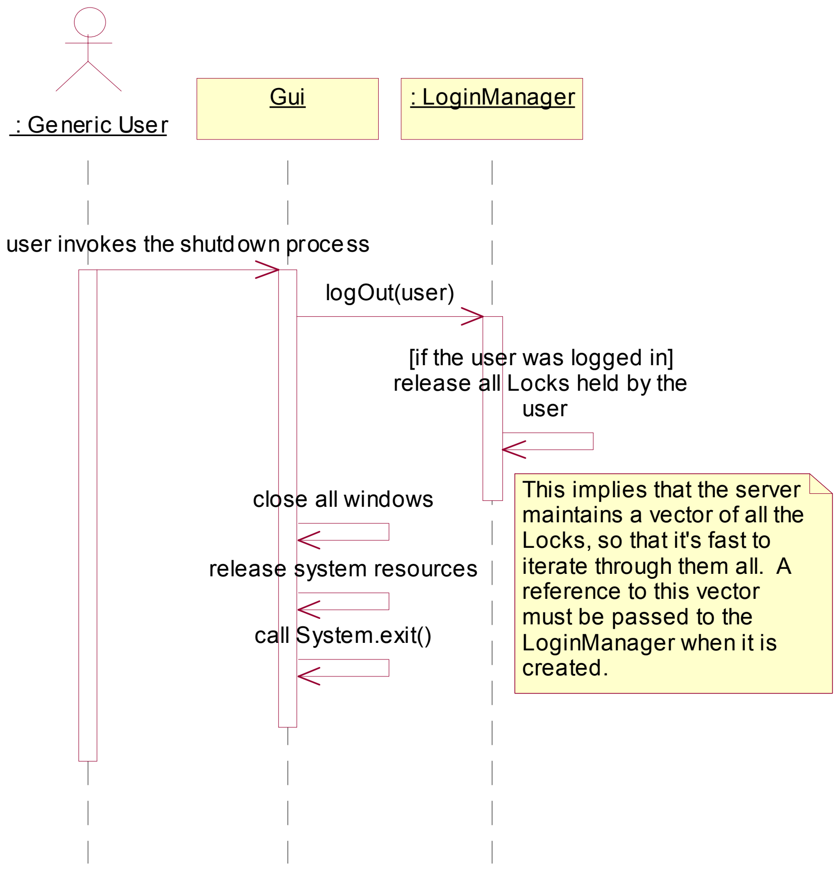

: Generic User
Gui
: LoginManager
user invokes the shutdown process
logOut(user)
[if the user was logged in]
release all Locks held by the user
close all windows
release system resources
call System.exit()
This implies that the server maintains a vector of all the Locks, so that it's fast to iterate through them all. A reference to this vector must be passed to the LoginManager when it is created.

Figure 34: Sequence Diagram for Select Item in Tree Use Case (Client Selected)

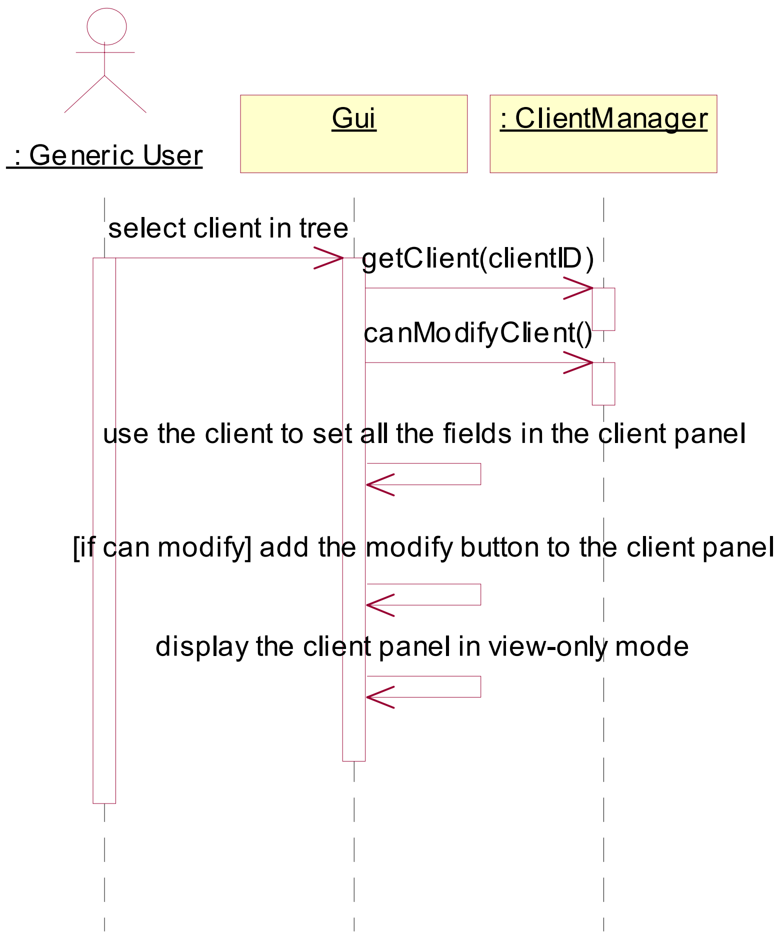

: Generic User
Gui
: ClientManager
select client in tree
getClient(clientID)
canModifyClient()
use the client to set all the fields in the client panel
[if can modify] add the modify button to the client panel
display the client panel in view-only mode

Figure 35: Sequence Diagram for Select Item in Tree Use Case (Project Selected)

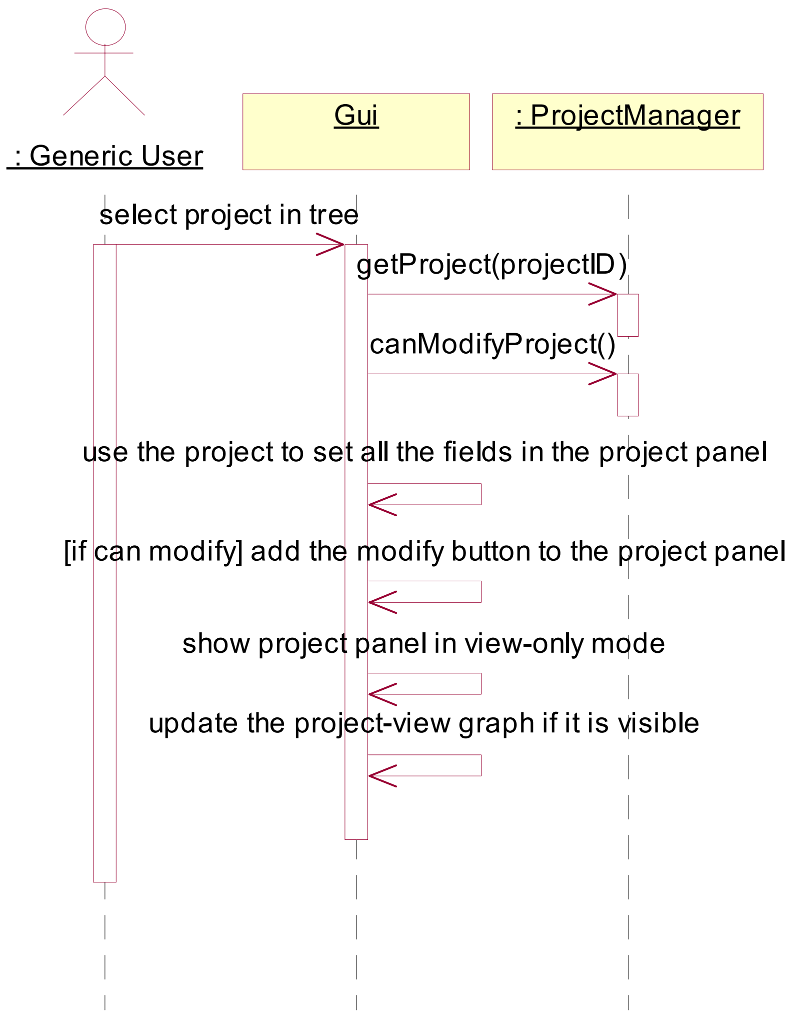

: Generic User
Gui
: ProjectManager
select project in tree
getProject(projectID)
canModifyProject()
use the project to set all the fields in the project panel
[if can modify] add the modify button to the project panel
show project panel in view-only mode
update the project-view graph if it is visible

Figure 36: Sequence Diagram for Select Item in Tree Use Case (Consultant Selected)

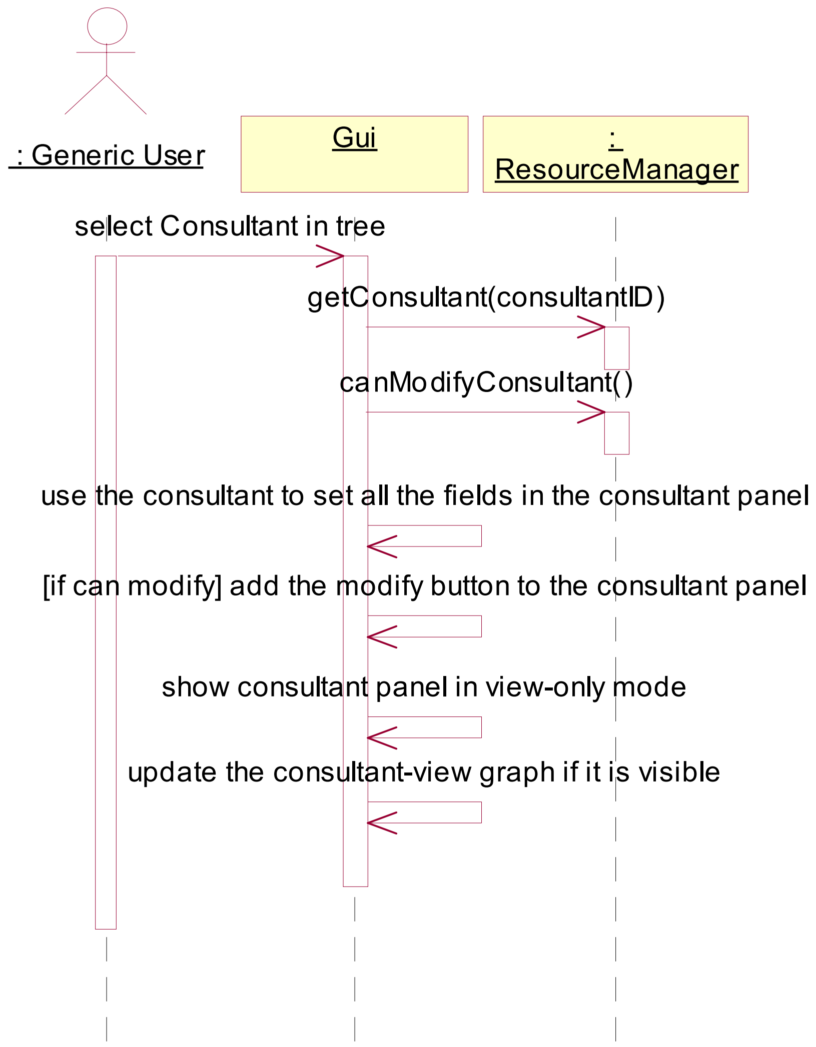

: Generic User
Gui
: ResourceManager
select Consultant in tree
getConsultant(consultantID)
canModifyConsultant()
use the consultant to set all the fields in the consultant panel
[if can modify] add the modify button to the consultant panel
show consultant panel in view-only mode
update the consultant-view graph if it is visible

Figure 37: Sequence Diagram for Select Item in Tree Use Case (Machine Selected)

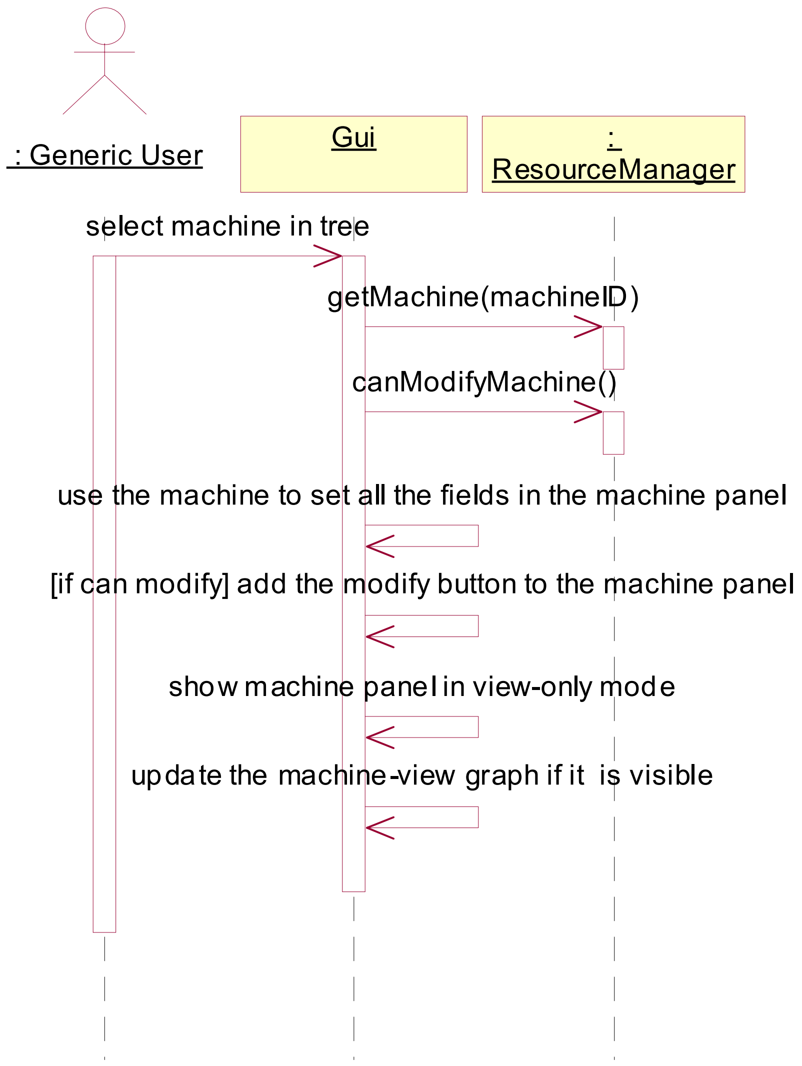

: Generic User
Gui
: ResourceManager
select machine in tree
getMachine(machineID)
canModifyMachine()
use the machine to set all the fields in the machine panel
[if can modify] add the modify button to the machine panel
show machine panel in view-only mode
update the machine-view graph if it is visible

Figure 38: Sequence Diagram for Select Item in Tree Use Case (Room Selected)

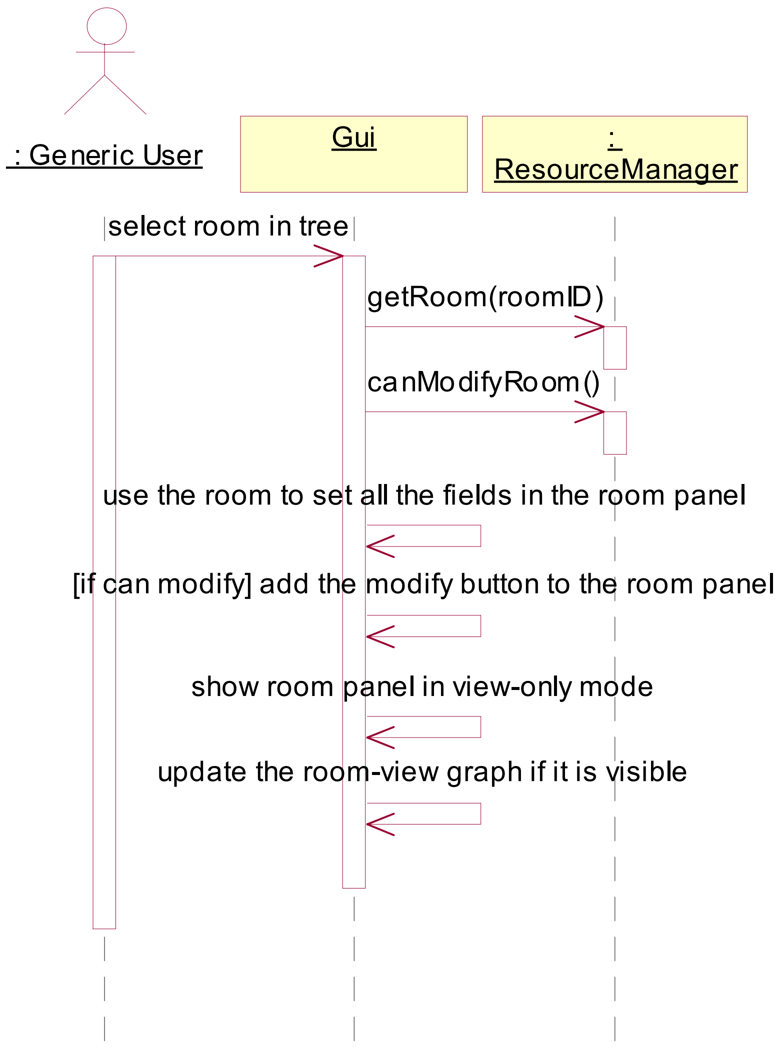

: Generic User
Gui
: ResourceManager
select room in tree
getRoom(roomID)
canModifyRoom()
use the room to set all the fields in the room panel
[if can modify] add the modify button to the room panel
show room panel in view-only mode
update the room-view graph if it is visible

Figure 39: Sequence Diagram for Select Item in Tree Use Case (ResourceAllocation Selected)

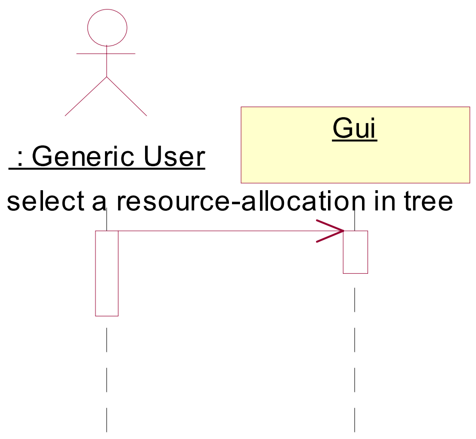
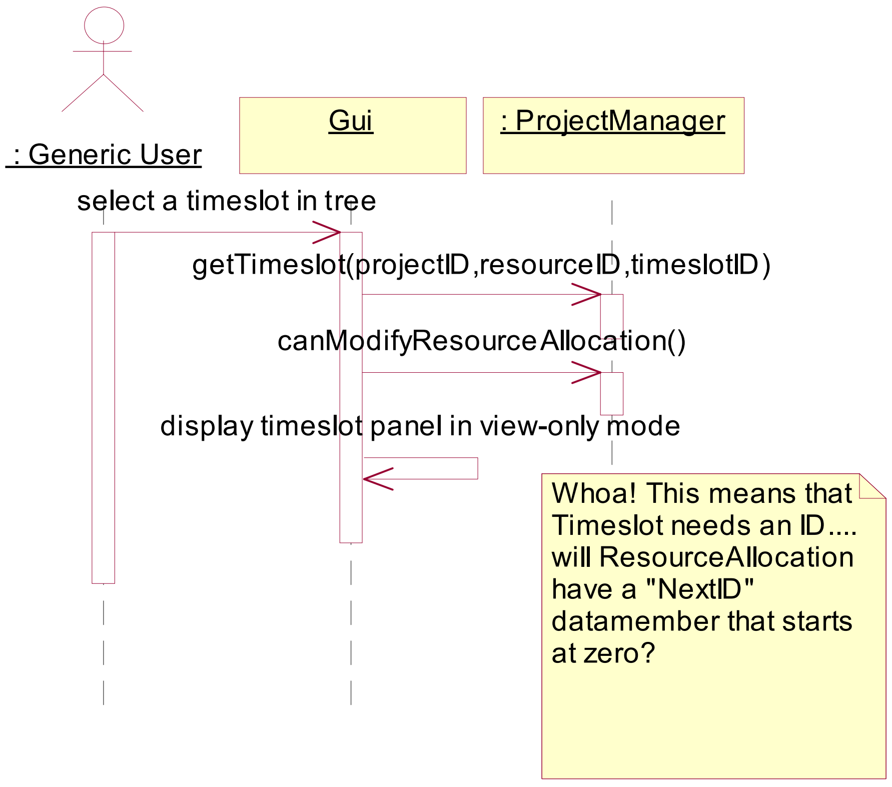


Figure 40: Sequence Diagram for Select Item in Tree Use Case (Timeslot Selected)


: Generic User
Gui
: ProjectManager
select a timeslot in tree
getTimeslot(projectID,resourceID,timeslotID)
canModifyResourceAllocation()
display timeslot panel in view-only mode
Whoa! This means that Timeslot needs an ID.... will ResourceAllocation have a "NextID" datamember that starts at zero?

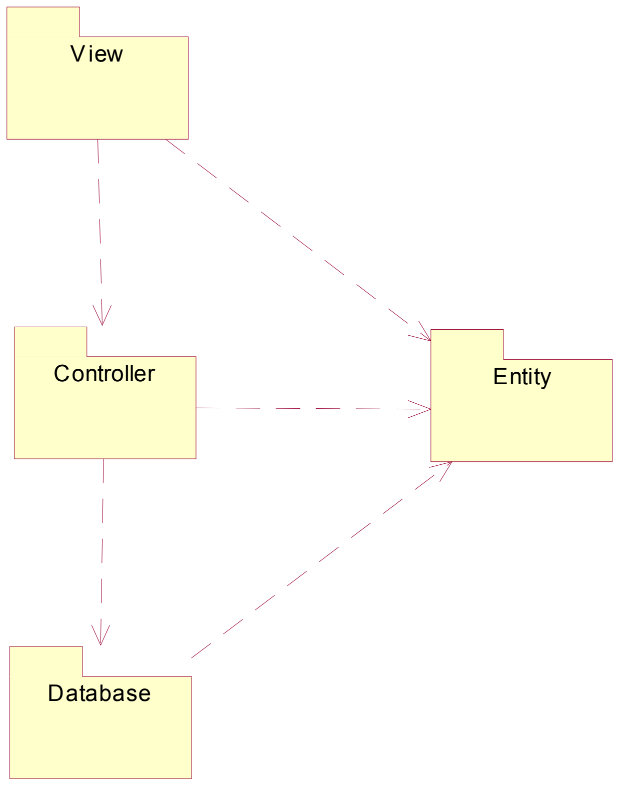
Class and Package Diagrams

Package Diagram

After creating all the sequence diagrams, a complete list of classes was developed. These classes were grouped into four packages. The View package contains all the classes that deal with the GUI, including dialog box, form, and graph classes. The Controller package contains classes that manipulate data and provide communication between the server and the clients. All of the manager classes are in this package. The Database package contains all the classes that directly access the database. The Entity package contains all of the data-rich classes that are stored in the database. These classes include Client, Resource, and Project.

A package diagram was constructed and can be found below. Note the lines indicate “knows” relationships. From the diagram, it can be seen that classes in the View package know about classes in the Controller package. (But classes in the Controller Package do not know about classes in the View package.) Also, the Controller package knows about the classes in the Database package. All of the packages know about the classes in the Entity package.

Figure 41: Package Diagram



Class Diagrams

Class diagrams show all the classes in the system. Class diagrams are structured like conceptual models but expand them in several ways. First, they add classes identified in the course of making the sequence diagrams. Second, they add methods (functions) to the classes. In addition to classes (including their attributes and methods), class diagrams display other information including associations between classes, interfaces, navigability, and dependencies (Larman 257-258).

The classes were divide among four different packages as explained above. A class diagram was made for each package excluding the Database package. (Since an object database is being used for persistent storage, the code required to store the data is trivial. Therefore, the no class diagram was made for the Database package.) Only those classes contained in a particular package are shown in the associated class diagram. The class diagrams constructed can be found below.

Figure 42: Class Diagram for the View Package

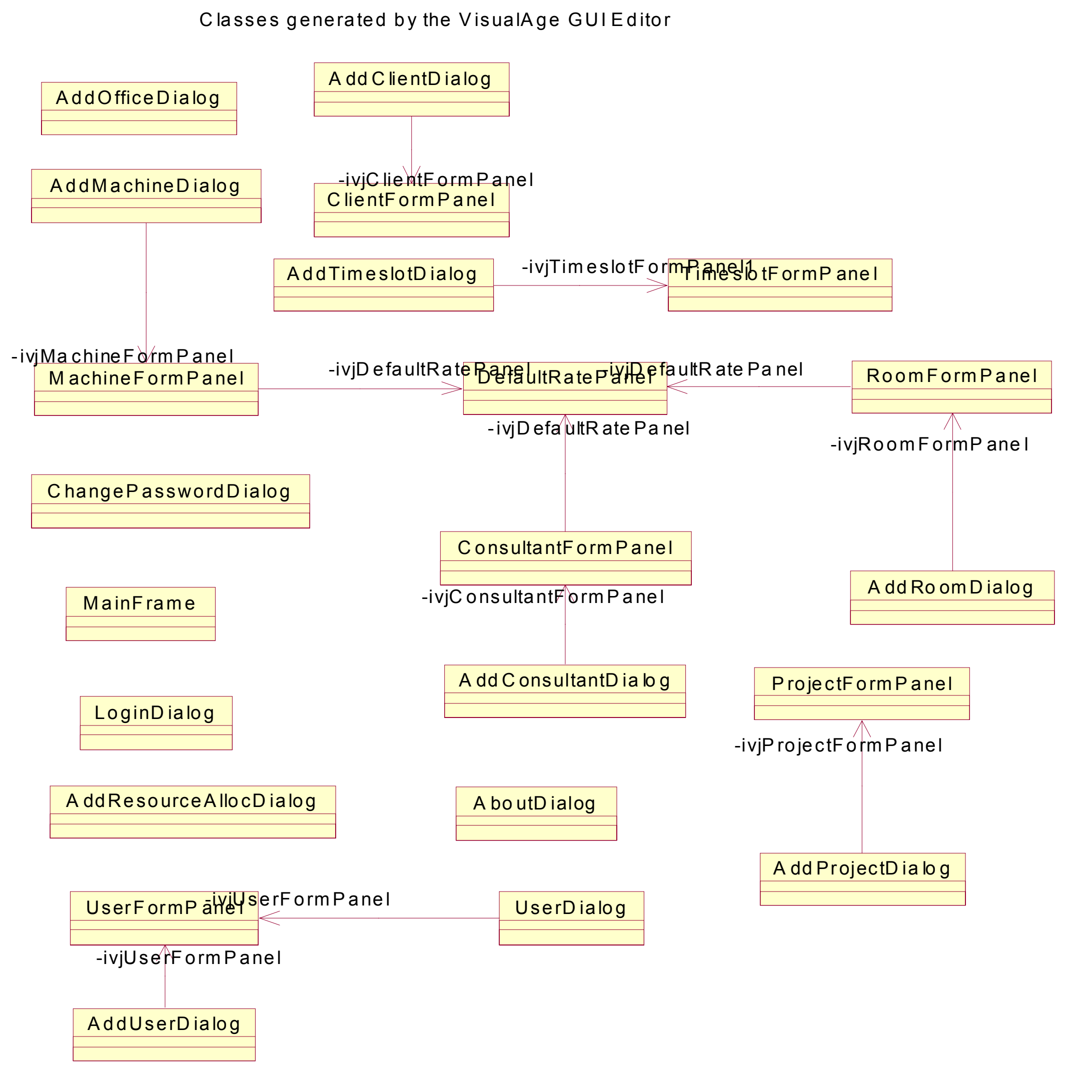


Figure 43: Class Diagram for the Entity Package

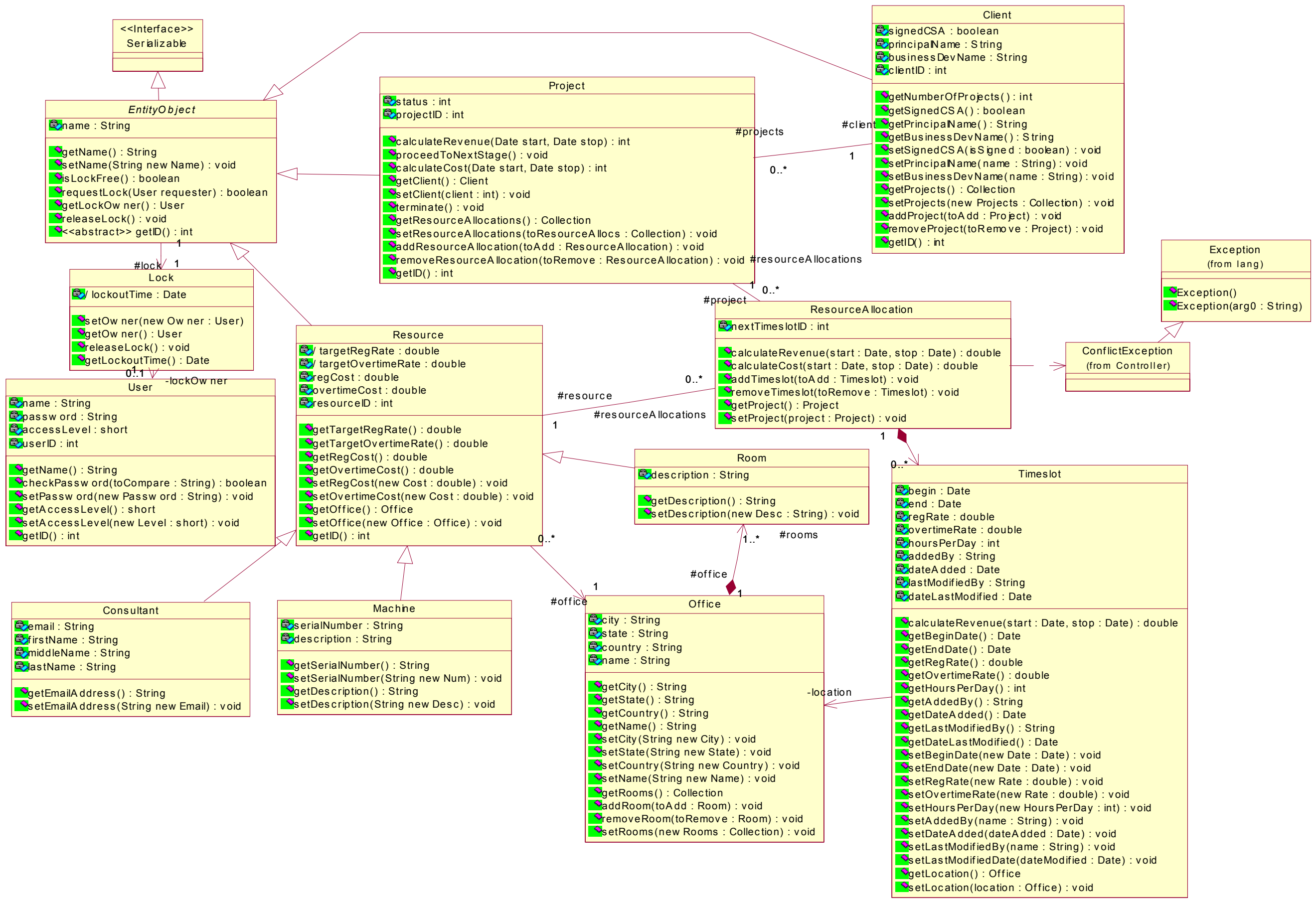
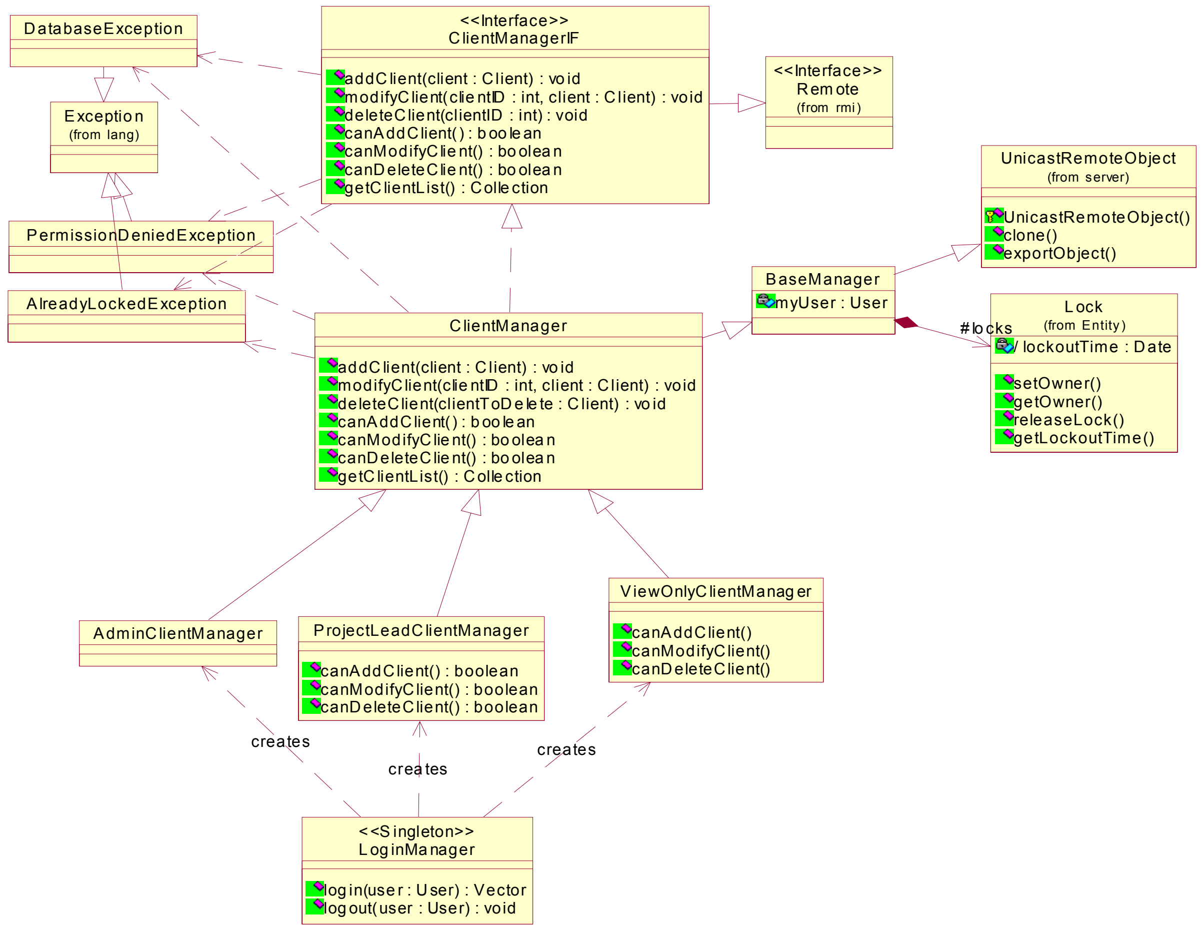

<<Interface>>
Serializable
EntityObject
name : String
getName() : String
setName(String new Name) : void
isLockFree() : boolean
requestLock(User requester) : boolean
getLockOwner() : User
releaseLock() : void
<<abstract>> getID() : int
Project
status : int
projectID : int
calculateRevenue(Date start, Date stop) : int
proceedToNextStage() : void
calculateCost(Date start, Date stop) : int
getClient() : Client
setClient(client : int) : void
terminate() : void
getResourceAllocations() : Collection
setResourceAllocations(toResourceAllocs : Collection) : void
addResourceAllocation(toAdd : ResourceAllocation) : void
removeResourceAllocation(toRemove : ResourceAllocation) : void
getID() : int
Client
signedCSA : boolean
principalName : String
businessDevName : String
clientID : int
getNumberOfProjects() : int
getSignedCSA() : boolean
getPrincipalName() : String
getBusinessDevName() : String
setSignedCSA(isSigned : boolean) : void
setPrincipalName(name : String) : void
setBusinessDevName(name : String) : void
getProjects() : Collection
setProjects(new Projects : Collection) : void
addProject(toAdd : Project) : void
removeProject(toRemove : Project) : void
getID() : int
#projects
#client
0..*
1
Exception
(from lang)
Exception()
Exception(arg0 : String)
ConflictException
(from Controller)
Lock
lockoutTime : Date
setOwner(new Owner : User)
getOwner() : User
releaseLock() : void
getLockoutTime() : Date
#lock
1
1
0..1
-lockOwner
User
name : String
password : String
accessLevel : short
userID : int
getName() : String
checkPassword(toCompare : String) : boolean
setPassword(new Password : String) : void
getAccessLevel() : short
setAccessLevel(new Level : short) : void
getID() : int
Resource
targetRegRate : double
targetOvertimeRate : double
regCost : double
overtimeCost : double
resourceID : int
getTargetRegRate() : double
getTargetOvertimeRate() : double
getRegCost() : double
getOvertimeCost() : double
setRegCost(new Cost : double) : void
setOvertimeCost(new Cost : double) : void
getOffice() : Office
setOffice(new Office : Office) : void
getID() : int
#resourceAllocations
1
0..*
#project
ResourceAllocation
nextTimeslotID : int
calculateRevenue(start : Date, stop : Date) : double
calculateCost(start : Date, stop : Date) : double
addTimeslot(toAdd : Timeslot) : void
removeTimeslot(toRemove : Timeslot) : void
getProject() : Project
setProject(project : Project) : void
#resource
0..*
#resourceAllocations
1
1
0..*
Room
description : String
getDescription() : String
setDescription(new Desc : String) : void
1..*
#rooms
#office
1
Timeslot
begin : Date
end : Date
regRate : double
overtimeRate : double
hoursPerDay : int
addedBy : String
dateAdded : Date
lastModifiedBy : String
dateLastModified : Date
calculateRevenue(start : Date, stop : Date) : double
getBeginDate() : Date
getEndDate() : Date
getRegRate() : double
getOvertimeRate() : double
getHoursPerDay() : int
getAddedBy() : String
getDateAdded() : Date
getLastModifiedBy() : String
getDateLastModified() : Date
setBeginDate(new Date : Date) : void
setEndDate(new Date : Date) : void
setRegRate(new Rate : double) : void
setOvertimeRate(new Rate : double) : void
setHoursPerDay(new HoursPerDay : int) : void
setAddedBy(name : String) : void
setDateAdded(dateAdded : Date) : void
setLastModifiedBy(name : String) : void
setLastModifiedDate(dateModified : Date) : void
getLocation() : Office
setLocation(location : Office) : void
Consultant
email : String
firstName : String
middleName : String
lastName : String
getEmailAddress() : String
setEmailAddress(String new Email) : void
Machine
serialNumber : String
description : String
getSerialNumber() : String
setSerialNumber(String new Num) : void
getDescription() : String
setDescription(String new Desc) : void
0..*
1
#office
Office
city : String
state : String
country : String
name : String
getCity() : String
getState() : String
getCountry() : String
getName() : String
setCity(String new City) : void
setState(String new State) : void
setCountry(String new Country) : void
setName(String new Name) : void
getRooms() : Collection
addRoom(toAdd : Room) : void
removeRoom(toRemove : Room) : void
setRooms(new Rooms : Collection) : void
-location

Figure 44: Class Diagram for the Controller Package



Class Descriptions

The class diagrams do a good job providing the details necessary to implement a class. However, it is difficult to look at the class diagrams and determine just how a particular class fits into the overall design. Thus, class descriptions are written to provide a text-based narrative of what the classes can do and how they will be used in the design. Below are descriptions for all of the important classes in the design.

User

An entity class that will contain a username, password, and full name for all users who can access data using the client program. The User class will also hold the access level, which will determine what manager functions they can call.

Entity Object

This is an abstract class that contains methods and attributes common to all objects that will be stored in the database. It includes a name attribute, a lock, and an ID number. Methods to manipulate the lock and name are provided. It serves as a base class for the classes Resource, Project and Client.

Project

An entity class that will contain all the information needed to describe a project. Note it has a container of ResourceAllocations, which holds references to all the Resources that can be used on a particular Project.

Client

An entity class that will contain all the information needed to describe a client. A container will be used to hold references to Projects assigned to a Client.

Resource

An abstract class that contains attributes and methods common to all resources. It contains a container that holds references to ResourceAllocations. These resource allocations indicate to which projects this particular resource is assigned. It is the parent class of Consultant, Room, and Equipment.

Machine

Derived from the Resource class. Describes an inanimate object that can be scheduled for use with a project. The class adds a serial number to the base class, which uniquely identifies the piece of equipment.

Room

Derived from the Resource class. Describes a single room in an Office that can be scheduled for use with a project. The class adds a room description attribute to the base class.

Consultant

Derived from the Resource class. Describes a consultant who can be scheduled to work on a project. The class adds an email address that will be used to uniquely identify each consultant.

ResourceAllocation

A class that holds data related to assignment of the resource to a particular project. The class contains a reference to the particular resource that has been assigned to a project.

Timeslot

The Timeslot class holds data pertaining to the scheduling of a resource. The class holds the start date and end date of the assignment, the number of hours per day the resource will be used, and the hourly rate the particular client will be charged for the use of the resource.

Lock

A class used to ensure that only one user at a time can modify an EntityObject. Prior to performing any operation that modifies data, a lock must be obtained. A lock can only be obtained if no other user has the lock checked out. The lock is released after the modification is performed.

LoginManager

This is the only class that will be accessible to clients via the naming service. It will have functions to validate users. After a valid username and password combination is supplied to the LoginManager, references to the other four managers will be passed to the client program. The type of managers passed back depends on the user's access level. This class must be remotely accessible.

ClientManager

This class is responsible for processing requests related to client data from the clients. This includes obtaining and changing Client objects. The ClientManager has a reference to the User who the object was created for, therefore it can be used to obtain locks when client data is to be modified. Three different classes will be derived from this class. There will be one derived class for each user access level in the system. Each derived class will provide differing levels of functionality. Functions will be provided in the base class that allow the client program to determine which functions it is allowed to perform. This class must be remotely accessible.

ResourceManager

This class is responsible for processing requests related to resource data from the clients. This includes obtaining and changing Resource objects. The ResourceManager has a reference to the User who the object was created for, therefore it can be used to obtain locks when resource data is to be modified. Three different classes will be derived from this class. There will be one derived class for each user access level in the system. Each derived class will provide differing levels of functionality. Functions will be provided in the base class that allow the client program to determine which functions it is allowed to perform. This class must be remotely accessible.

ProjectManager

This class is responsible for processing requests related to project data from the clients. This includes obtaining and changing Project objects, ResourceAllocation objects, and Timeslot objects. The ProjectManager has a reference to the User who the object was created for, therefore it can be used to obtain locks when an object needs to be modified. Three different classes will be derived from this class. There will be one derived class for each user access level in the system. Each derived class will provide differing levels of functionality. Functions will be provided in the base class which allow the client program to determine which functions it is allowed to perform. This class must be remotely accessible.

MiscManager

This class is responsible for processing miscellaneous requests from clients. It will perform such tasks as changing passwords and adding, modifying, and deleting users, and generating graphs. Three different classes will be derived from this class. There will be one derived class for each user access level in the system. Each derived class will provide differing levels of functionality. Functions will be provided in the base class which allow the client program to determine which functions it is allowed to perform. This class must be remotely accessible.

System Contracts

System operation contracts describe changes in the state of the system when an operation is performed. Contracts are normally written for each method (function) present in a design. Contracts are declarative— they describe what will happen when a method is called, not how it will happen (Larman 147).

System contracts were written for all non-trivial methods contained in the Entity and Controller packages. No system contracts were written for trivial functions, including accessors and mutators that don't exhibit any special behavior. Also, no system contracts were written for any of the functions in the View package. The methods for the classes in this package were generated by IBM Visual Age based on a graphical layout of the forms and dialog boxes. The majority of these generated methods will not be used, so contracts are not needed.

A simplified version of system contracts was used. The method signature, a description of what the method does, preconditions (necessary conditions for the proper operation of the method), and postconditions (changes in the state of the overall system after the method completes execution) are contained in the system contracts below. Information related to method type, responsibilities, and output (which Larman suggested be in the contracts) was considered unnecessary and was excluded.

System Contracts for Classes in the Entity Package

EntityObject

getName () : String
setName (String : newName) : void
isLockFree () : boolean

Description: Checks to see if the lock associated with the EntityObject is not in use (is equal to null).
Preconditions: None.
Postconditions: None.

requestLock (User : requester) : boolean

Description: Reserves the lock by calling setOwner() on the Lock object Returns true if the lock was free and the lock was reserved. Returns false if the lock was in use and the request couldn't be fulfilled.
Preconditions: None.
Postconditions: Lock is reserved for User if it was free. Otherwise, nothing occurs.

getLockOwner () : User

Description: Returns a reference to the User holding the Lock for the EntityObject. Calls getUser() on the Lock contained inside the EntityObject.
Preconditions: None.
Postconditions: The User holding the lock is returned. Null is returned if the lock is free.

releaseLock () : void

Description: Releases the lock for the EntityObject by calling the releaseLock() function on the Lock object contained inside the EntityObject.
Preconditions: None.
Postconditions: Releases the lock.

Client

getNumberOfProjects () : int
Description: Counts the number of projects listed to the client.
Preconditions: The client has been created
Postconditions: The number of projects is returned
getSignedCSA () : boolean
getPrincipalName () : String
getBusinessDevName () : String
setSignedCSA (isSigned : boolean) : void
setPrincipalName (name : String) : void
setBusinessDevName (name : String) : void
getProjects () : Collection
Description: Returns all the projects for a particular client.
Preconditions: There are projects for the client.
Postconditions: The projects for the client are returned
setProjects (newProjects : Collection) : void
Description: Sets the list of projects for the client
Preconditions: There are no projects for the particular client
Postconditions: The client now has a list of projects
addProject (toAdd : Project) : void
Description: Adds a project to the client.
Preconditions: The list of projects has already been created
Postconditions: A new project is associated with that client
removeProject (toRemove : Project) : void
Description: Removes a project from the list of projects for the client
Preconditions: The project to be removed exists.
Postconditions: The project is removed from the client.

Consultant

getEmailAddress () : String
setEmailAddress (newEmail : String) : void

Lock

setOwner (newOwner : User) : void
Description: Set which user is the owner of the lock. It also sets the lockoutTime, the time that the user acquired the lock.
Preconditions: None
Postconditions: The user is now listed as the lock owner.
getOwner () : User
getLockoutTime () : Date
releaseLock () : void
Description: Sets the owner of the lock to null.
Preconditions: There was an owner of the lock.
Postcontitions: The lock has no owner. (It is null)

Machine

getSerialNumber () : String
setSerialNumber (newNum : String) : void
getDescription () : String

Description: Set the description of the machine.
Preconditions: The machine exists and the description exists.
Postconditions: A description is associated with a machine.

setDescription (newDesc : String) : void

Description: Returns the description of the machine.
Preconditions: The machine exists and a description is passed in.
Postconditions: The description of the machine is returned.

Office

getCity () : String
getState () : String
getCountry () : String
getName () : String
setCity (newCity : String) : void
setState (newState : String) : void
setCountry (newCountry : String) : void
setName (newName : String) : void
getRooms () : Collection

Description: Finds all of the rooms in this office and returns them as a collection of rooms.
Preconditions: At least one room exists in this office.
Postconditions: The collection of rooms is returned.

addRoom (toAdd : Room) : void

Description: Adds the room to the collections of rooms that already exist in this office.
Preconditions: The room must not already exist in the collection of existing rooms.
Postconditions: The room is added to the collection of existing rooms.

removeRoom (toRemove : Room) : void

Description: Deletes the room (given in parameter) from the list of rooms, in this office.
Preconditions: The room must exist in the collection of rooms, in this office.
Postconditions: The room (given in parameter) is removed from the collection of rooms in this office.

setRooms (newRooms : Collection) : void

Description: Adds the passed in collection of rooms to the office. If any rooms already exist in the office, nothing happens (no rooms are added).
Preconditions: The currently existing collection of rooms must be null before the user calls this function. Otherwise, no rooms will be added.
Postconditions: The whole collections of passed in rooms will be added to the office.

Project

calculateRevenue (start : Date, stop : Date) : int

Description: Calculates the income for a particular project in the date range provided. Uses the actual rate from each Timeslot multiplied by the number of hours used to calculate the income.
Preconditions: start Date less than or equal to (comes before) stop Date.
Postconditions: The income for the project for the date range (in cents) is returned.

proceedToNextStage () : void

Description: Increments the status of a project, moving it to the next stage. If the project is already in the terminated stage, nothing happens.
Preconditions: None.
Postconditions: The project status is moved to the next stage unless it was already in the final stage (terminated).

calculateCost (start : Date, stop : Date) : Double

Description: Calculates the cost of all the resources used on a project for the given date range. Uses the cost and overtime cost from all the Resources allocated to the project along with the hours per day specified in the Timeslots to perform the calculation.
Preconditions: start Date less than or equal to (comes before) stop Date.
Postconditions: The cost of the project for the date range (in cents) is returned.

getClient() : Client

setClient(newClient : Client) : void

terminate () : void

Description: Sets the status of a Project to terminated.
Preconditions: None.
Postconditions: The status of the Project is set to terminated if it isn't already in that state.

getResourceAllocations () : Collection

Description: Returns a collection containing all the ResourceAllocations contained in the Project.
Preconditions: None.
Postconditions: A collection is returned that contains the ResourceAllocations for the Project.

setResourceAllocations (toResourceAllocs : Collection) : void

Description: Allows a collection of ResourceAllocations to be added to the Project. Note this collection is not appended to the collection maintained by the Project; instead, this collection is completely replaced.
Preconditions: the collection of ResourceAllocations in the project is null (has had no ResourceAllocations) added and the project is not in the terminated state.
Postconditions: The collection of ResourceAllocations in the Project is the same as the collection passed in.

addResourceAllocation (toAdd : ResourceAllocation) : void

Description: Adds a ResourceAllocation to the collection held by the Project.
Preconditions: The Project is not in the terminated state.
Postconditions: The ResourceAllocation is added to the collection in the Project.

removeResourceAllocation (toRemove : ResourceAllocation) : void

Description: Removes a ResourceAllocation from the collection of ResourceAllocations held by the client. A ResourceAllocation is only removed if the reference passed in matches one in the collection.
Preconditions: At least one ResourceAllocation exists in the collection stored in the Project.
Postconditions: If a match is found, the specified ResourceAllocation is removed from the collection in the Project. Otherwise, nothing happens.

ResourceAllocation

calculateRevenue (start : Date, stop : Date) : Double

Description: Calls all of the "calculateRevenue" functions of all of the Timeslots that this resource allocation contains.
Preconditions: Start date must be physically before stop date. Access level must be high enough to call this function.

Postconditions: The calculated amount for the revenue, of all of the appropriate timeslots, is returned.

calculateCost (start : Date, stop : Date) : Double

Description: Goes to the resource and calls “getRegCost” and “getOvertimeCost” functions, and look at how many hours per week the resource is being used (less than 40 = regCost, over 40 = overtimeCost), and calculate the cost for the resource between the start date and stop date.
Preconditions: Start date must be physically before stop date. Access level must be high enough to call this function.
Postconditions: The cost is returned

addTimeslot (toAdd : Timeslot) : void

Description: Adds the passed in timeslot to the collection of timeslots which already exist.
Preconditions: Access level must be high enough to call this function
Postconditions: The timeslot is added to the collection of timeslots which already exist.

removeTimeslot (toRemove : Timeslot) : void

Description: Removes the timeslot from the collection of timeslots which already exist.
Preconditions: The timeslot to be removed much be in the collection of timeslots. Access level much be high enough to call this function.
Postconditions: The passed in timeslot is removed.

setProject(project : Project) : void

getProject() : Project

Timeslot

calculateRevenue (start : Date, stop : Date) : Double

Description: Calculates the revenue of the resource for this particular timeslot by using the rates, hours per day and the days allocated for the time period inputted.
Preconditions: The stop date is after the start date.The start date is before end Date of the timeslot. The stop date is after the start Date of the timeslot.
Postconditions: The revenue of the resource is returned.

getBeginDate () : Date

getEndDate () : Date

getRegRate () : Double

getOvertimeRate () : Double

getHoursPerDay () : int

getAddedBy () : String

Description: Returns who added the timeslot to the resource allocation.
Preconditions: The added by field was set.
Postconditions: The users name who added the timeslot was returned.

getDateAdded () : Date

getLastModifiedBy () : String

Description: Returns the name of the user who last modified the timeslot.
Preconditions: The user name was set.
Postconditions: The user who last modified the timeslot was returned.

getDateLastModified () : Date

setBeginDate (newDate : Date) : void

setEndDate (newDate : Date) : void

setRegRate (newRate : Double) : void

setOvertimeRate (newRate : Double) : void

setHoursPerDay (newHoursPerDay : int) : void

setAddedBy (name : String) : void
Description: Set the name of the user who added the timeslot to the resource allocation.
Preconditions: None.
Postconditions: The string has been set with the users name.

setDateAdded (dateAdded : Date) : void

setLastModifiedBy (name : String) : void
Description: Set the name of the user who was the last person to make modifications to the timeslot.
Preconditions: None
Postconditions: The name of the last person to modify this timeslot is set.

setLastModifiedDate (dateModified : Date) : void

getLocation () : Office

setLocation (location : Office) : void

System Contracts for Classes in the Controller Package

ProjectManager

canModifyProject () : boolean
Description: Returns true if the user has access to modify Projects. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

canAddProject () : boolean
Description: Returns true if the user has access to add Projects. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

canDeleteProject () : boolean
Description: Returns true if the user has access to modify Projects. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

addProject (project : Project, clientID : int) : void
Description: Adds the Project passed in to the given Client. Also, adds the Project to the database. Calls canAddProject() in order to determine if an exception should be thrown or not.
Preconditions: The clientID passed in is valid.
Postconditions: The Project is added to the container inside the given Client.

canAddResourceAllocation () : boolean
Description: Returns true if the user has access to add ResourceAllocations. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

canAMDTimeslot () : boolean
Description: Returns true if the user has access to add, modify, and delete Timeslots. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

canDeleteResourceAllocation () : boolean

Description: Returns true if the user has access to delete ResourceAllocations. Otherwise returns false. In the base class (ProjectManager) true is always returned.
Preconditions: None.
Postconditions: None.

modifyProject (project : Project, projectID : int) : void

Description: Modifies the Project with the given ID number. Updates the database with the new information. Calls canModifyProject() in order to determine if an exception should be thrown or not.
Preconditions: The projectID passed in is valid.
Postconditions: Modifies the Project with the given ID number. The Project is modified so its data members match those in the Project passed in.

deleteProject (projectID : int) : void

Description: Deletes the Project with the given ID number. If the Project has any resource allocations, it is put in the terminated state and stays in the database. Otherwise, if it has no ResourceAllocations, it is removed from the database. Calls canDeleteProject() in order to determine if an exception should be thrown or not.
Preconditions: The projectID passed in is valid.
Postconditions: The Project is “deleted”; its state is either set to terminated or it is removed from the database.

addResourceAllocation (projectID : int, resourceID : int) : void

Description: First, a ResourceAllocation object is created using the resourceID passed in. This ResourceAllocation is then added to collections in the associated Resource and Project. Finally, the ResourceAllocation is added to the database.
Preconditions: The projectID and resourceID passed in are valid.
Postconditions: A ResourceAllocation is added to a Project, a Resource, and the database.

deleteResourceAllocation (projectID : int, resourceID : int) : void

Description: Deletes a ResourceAllocation by removing it from the
Preconditions: The projectID and resourceID passed in are valid.
Postconditions: A ResourceAllocation is added to a Project, a Resource, and the database.

addTimeslot (projectID : int, resourceID : int, timeslot : Timeslot) : void

Description: Adds a Timeslot to a ResourceAllocation. Uses the projectID and resourceID passed in to determine the proper ResourceAllocation. The Timeslot is also added to the database. Calls canAMDTimeslot()in order to determine if an exception should be thrown or not.
Preconditions: The projectID and resourceID passed in are valid.
Postconditions: The Timeslot is added to the proper ResourceAllocation and to the database.

modifyTimeslot (projectID : int, resourceID : int, timeslotID : int, timeslot : Timeslot) : void

Description: Modifies the Timeslot with the given ID with the information passed in. The projectID and resourceID are used to determine which ResourceAllocation contains the Timeslot. Calls canAMDTimeslot()in order to determine if an exception should be thrown or not.
Preconditions: The projectID, resourceID, and timeslotID passed in are valid.
Postconditions: The given Timeslot is updated using the information passed in.

deleteTimeslot (projectID : int, resourceID : int, timeslotID : int) : void

Description: Deletes the Timeslot with the given ID. Removes the Timeslot from the ResourceAllocation and from the database. The projectID and resourceID are used to determine which ResourceAllocation contains the Timeslot. Calls canAMDTimeslot() in order to determine if an exception should be thrown or not.
Preconditions: The projectID, resourceID, and timeslotID passed in are valid.
Postconditions: The given Timeslot is deleted.

getProject (projectID : int) : Project
Description: Returns a reference to the Project with the given ID.
Preconditions: The projectID provided is valid.
Postconditions: A reference to the given project ID is returned.

getResourceAllocation (projectID : int, resourceID : int) : ResourceAllocation
Description: Returns a reference to the ResourceAllocation designated by the projectID and the resourceID.
Preconditions: The projectID provided is valid.
Postconditions: A reference to the specified ResourceAllocation is returned.

getTimeslot (projectID : int, resourceID : int, timeslotID : int) : Timeslot
Description: Returns a reference to the Timeslot with the given ID. The projectID and resourceID are used to determine which ResourceAllocation contains the Timeslot.
Preconditions: The projectID, resourceID, and timeslotID provided are valid.
Postconditions: A reference to the specified Timeslot is returned.

obtainProjectLock (projectID : int) : void
Description: Locks the Project with the given ID number. Uses the reference to the User contained in the ProjectManager class. Calls requestLock() on the Project. Adds the Lock to the container of Locks held by the ProjectManager. Throws an exception if the lock cannot be obtained.
Preconditions: The projectID provided is valid.
Postconditions: Locks the Project if possible.

releaseProjectLock (projectID : int) : void
Description: Releases the lock on the Project with the given ID number. Calls releaseLock() on the Project. Removes the Lock from the container of Locks held by the ProjectManager.
Preconditions: The projectID provided is valid.
Postconditions: Releases the lock on the specified Project.

GetProjectList (clientID : int) : Collection
Description: Returns a collection containing the name and ID all the Projects associated with the given client.
Preconditions: The clientID passed in is valid.
Postconditions: A collection containing the name and ID of all the Projects assigned to the Client with the ID passed in.

getResourceAllocationList (projectID : int) : Collection
Description: Returns a collection containing the name and ID all the ResourceAllocations associated with the given Project.
Preconditions: The projectID passed in is valid.
Postconditions: A collection containing the name and ID of all the ResourceAllocations associated with the Project with the ID passed in.

getTimeslotList (projectID : int, resourceID : int) : Collection
Description: Returns a collection containing the name and ID all the Timeslots associated with the given ResourceAllocation. The projectID and resourceID are used to determine which ResourceAllocation to look at.
Preconditions: The projectID and resourceID passed in are valid.
Postconditions: A collection containing the name (start time and stop time) and ID of all the Timeslots associated with the ResourceAllocation passed in.

ProjectLeadProjectManager

canAddProject () : boolean
Description: Returns false because users with View Only access cannot add Projects.
Preconditions: None.
Postconditions: Returns false.

canModifyProject () : boolean
Description: Returns false because users with View Only access cannot modify Projects.
Preconditions: None.
Postconditions: Returns false.

canDeleteProject () : boolean
Description: Returns false because users with View Only access cannot delete Projects.
Preconditions: None.
Postconditions: Returns false.

canAddResourceAllocation () : boolean
Description: Returns false because users with View Only access cannot add ResourceAllocations.
Preconditions: None.
Postconditions: Returns false.

canDeleteResourceAllocation () : boolean
Description: Returns false because users with View Only access cannot delete ResourceAllocations.
Preconditions: None.
Postconditions: Returns false.

ViewOnlyProjectManager

canAddProject () : boolean
Description: Returns false because users with View Only access cannot add Projects.
Preconditions: None.
Postconditions: Returns false.

canModifyProject () : boolean
Description: Returns false because users with View Only access cannot modify Projects.
Preconditions: None.
Postconditions: Returns false.

canDeleteProject () : boolean
Description: Returns false because users with View Only access cannot delete Projects.
Preconditions: None.
Postconditions: Returns false.

canAddResourceAllocation () : boolean
Description: Returns false because users with View Only access cannot add ResourceAllocations.
Preconditions: None.
Postconditions: Returns false.

canAMDTimeslot () : boolean
Description: Returns false because users with View Only access cannot add, modify, or delete Timeslots.
Preconditions: None.
Postconditions: Returns false.

canDeleteResourceAllocation () : boolean
Description: Returns false because users with View Only access cannot delete ResourceAllocations.
Preconditions: None.
Postconditions: Returns false.

getTimeslot(projectID : int, resourceID : int, timeslotID : int) : Timeslot
Description: Returns the Timeslot that matches the arguments passed in with confidential (rate) information removed.
Preconditions: The projectID, resourceID, and timeslotID passed in are valid.
Postconditions: Returns the given Timeslot with the confidential information removed.

ResourceManager

modifyConsultant (consultantID : int, consultant : Consultant) : void
Description: Changes the consultant information in the database subsystem.
Preconditions: The information existed.
Postconditions: The consultant information is changed.

modifyMachine (machineID : int, machine : Machine) : void
7Description: Changes the machine information in the database subsystem.
Preconditions: The information existed.
Postconditions: The machine information is changed.

modifyRoom (roomID : int, room : Room) : void
Description: Changes the room information in the database subsystem.
Preconditions: The information existed.
Postconditions: The room information is changed.

canModifyConsultant () : boolean
Description: Checks the access level of the user to see if they can change the consultant information. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean to indicate if they can change the consultant information.

canModifyMachine () : boolean
Description: Checks the access level of the user to see if they can change the machine information. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canModifyRoom () : boolean
Description: Checks the access level of the user to see if they can change the room information. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canAddConsultant () : boolean
Description: Checks the access level of the user to see if they can add a consultant. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canAddMachine () : boolean
Description: Checks the access level of the user to see if they can add a machine. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canAddRoom () : boolean
Description: Checks the access level of the user to see if they add a room. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canDeleteConsultant () : boolean
Description: Checks the access level of the user to see if they can delete a consultant. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canDeleteMachine () : boolean
Description: Checks the access level of the user to see if they can delete a machine. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

canDeleteRoom () : boolean
Description: Checks the access level of the user to see if they can delete a room. In the base class (ResourceManager) true is always returned.
Preconditions: None
Postconditions: Returns a boolean indicating the outcome of the check.

getConsultant (consultantID : int) : Consultant
Description: Goes to the database subsystem and gets the specified consultant.
Preconditions: None
Postconditions: Returns the requested consultant.

getMachine (machineID : int) : Machine
Description: Goes to the database subsystem and gets the specified machine.
Preconditions: None
Postconditions: Returns the requested machine.

getRoom (roomID : int) : Room
Description: Goes to the database subsystem and gets the specified room.
Preconditions: None
Postconditions: Returns the requested room.

deleteConsultant (consultantID : int) : void
Description: Removes the specified consultant from the database subsystem.
Preconditions: There are no resource allocations for the consultant.
Postconditions: The consultant no longer is in the database subsystem.

deleteMachine (machineID : int) : void
Description: Removes the specified machine from the database subsystem.
Preconditions: There are no resource allocations for the machine.
Postconditions: The machine no longer exists in the database subsystem.

deleteRoom (roomID : int) : void
Description: Removes the specified room from the database subsystem.
Preconditions: There are no resource allocations for the room.
Postconditions: The room no longer exists in the database subsystem.

addConsultant (consultant : Consultant) : void
Description: Creates a new consultant in the database subsystem with the specified information.
Preconditions: None
Postconditions: The consultant is added to the database subsystem.

addMachine (machine : Machine) : void
Description: Creates a new machine in the database subsystem with the specified information.
Preconditions: None
Postconditions: The machine is added to the database subsystem.

addRoom (room : Room) : void
Description: Creates a new room in the database subsystem with the specified information.
Preconditions: None
Postconditions: The room is added to the database subsystem.

obtainResourceLock (resourceID : int) : void

Description: Locks the resource with the given ID number. Uses the reference to the User contained in the ResourceManager class. Calls requestLock() on the Resource. Throws an exception if the lock could not be obtained.
Preconditions: None
Postconditions: Locks the resource if possible.

releaseResourceLock (resourceID : int) : void

Description: Unlocks the resource with the given ID number. Uses the reference to the User contained in the ResourceManager class. Calls releaseLock() on the Resource.
Preconditions: A lock on the resource exists.
Postconditions: Unlocks the Resource.

getConsultantList () : Collection

Description: Goes to the database subsystem and returns a collection of consultants.
Preconditions: The database contains at least one consultant.
Postconditions: The collection of consultants is returned.

getRoomList () : Collection

Description: Goes to the database subsystem and returns a collection of rooms.
Preconditions: The database contains at least one room.
Postconditions: The collection of rooms is returned.

getMachineList () : Collection

Description: Goes to the database subsystem and returns a collection of machines.
Preconditions: The database contains at least one machine.
Postconditions: The collection of machines is returned.

ProjectLeadResourceManager

canModifyConsultant () : boolean

Description: Checks the access level to see if the specified user can modify a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating that the consultant cannot be modified.

canModifyMachine () : boolean

Description: Checks the access level to see if the specified user can modify a machine and returns false.
Preconditions: None
Postconditions: Returns false indicating that the machine cannot be modified.

canModifyRoom () : boolean

Description: Checks the access level to see if the specified user can modify a room and returns false.
Preconditions: None
Postconditions: Returns false indicating that the room cannot be modified.

canAddConsultant () : boolean

Description: Checks that access level to see if the specified user can add a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a consultant.

canAddMachine () : boolean

Description: Checks that access level to see if the specified user can add a machine and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a machine.

canAddRoom () : boolean

Description: Checks that access level to see if the specified user can add a room and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a room.

canDeleteConsultant () : boolean

Description: Checks that access level to see if the specified user can delete a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot delete a consultant.

ViewOnlyResourceManager

canModifyConsultant () : boolean

Description: Checks the access level to see if the specified user can modify a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating that the consultant cannot be modified.

canModifyMachine () : boolean

Description: Checks the access level to see if the specified user can modify a machine and returns false.
Preconditions: None
Postconditions: Returns false indicating that the machine cannot be modified.

canModifyRoom () : boolean

Description: Checks the access level to see if the specified user can modify a room and returns false.
Preconditions: None
Postconditions: Returns false indicating that the room cannot be modified.

canAddConsultant () : boolean

Description: Checks that access level to see if the specified user can add a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a consultant.

canAddMachine () : boolean

Description: Checks that access level to see if the specified user can add a machine and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a machine.

canAddRoom () : boolean

Description: Checks that access level to see if the specified user can add a room and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot add a room.

canDeleteConsultant () : boolean

Description: Checks that access level to see if the specified user can delete a consultant and returns false.
Preconditions: None
Postconditions: Returns false indicating the specified user cannot delete a consultant.

getConsultant (consultantID : int) : Consultant

Description: Goes to the database subsystem and gets the specified consultant. Deletes all confidential information, which includes cost and overtime cost.
Preconditions: None
Postconditions: Returns the requested consultant with confidential information removed.

getMachine (machineID : int) : Machine

Description: Goes to the database subsystem and gets the specified machine. Deletes all confidential information, which includes cost and overtime cost.
Preconditions: None
Postconditions: Returns the requested consultant with confidential information removed.

getRoom (roomID : int) : Room

Description: Goes to the database subsystem and gets the specified room. Deletes all confidential information, which includes cost and overtime cost.
Preconditions: None
Postconditions: Returns the requested consultant with confidential information removed.

ClientManager

addClient (client : Client) : void

Description: Adds the client to the collection of clients in the database.
Preconditions: Client must not already exist in database
Postconditions: Client is added to the database

modifyClient (clientID : int, client : Client) : void

Description: Finds the client in the database, which has the specified clientID, and replaces that client with the passed in client.
Preconditions: Client exists in the database (e.g. clientID is found).
Postconditions: The specified client data is updated to the info. from the passed in client.

deleteClient (clientToDelete : Client) : void

Description: Finds the client in the database, which has the specified clientID, and removes it from the collection of clients.
Preconditions: Client exists in the database.
Postconditions: Client is deleted from the collection of clients in the database.

canAddClient () : boolean

Description: Returns “true”. However, when this class is sub-classed, that class will override this function to return “false” if the user which uses that class does not have the rights to add a client.

canModifyClient () : boolean

Description: Returns “true”. However, when this class is sub-classed, that class will override this function to return “false” if the user which uses that class does not have the rights to modify a client.

canDeleteClient () : boolean

Description: Returns “true”. However, when this class is sub-classed, that class will override this function to return “false” if the user which uses that class does not have the rights to delete a client.

getClientList () : Collection

Description: Returns the whole list of clients that is stored in the database.

ProjectLeadClientManager

canAddClient () : boolean

Description: Returns “false” because the Project Lead user cannot add a client.

canModifyClient () : boolean

Description: Returns “false” because the Project Lead user cannot modify a client.

canDeleteClient () : boolean

Description: Returns “false” because the Project Lead user cannot delete a client.

ViewOnlyClientManager

canAddClient () : boolean

Description: Returns “false” because the View Only user cannot add a client.

canModifyClient () : boolean

Description: Returns “false” because the View Only user cannot modify a client.

canDeleteClient () : boolean

Description: Returns “false” because the View Only user cannot delete a client.

LoginManager

login (user : User) : Vector

This method returns a Vector that contains three items.
1.) A remote-reference to the ClientManager. (index 0)
2.) A remote-reference to the ProjectManager. (index 1)
3.) A remote-reference to the ResourceManager. (index 2)
Description: Logs in the specified user if username and password is correct. First, the passed in user is found in the database (if it’s in there), then it compares the passed in user’s password with the database user’s password. If passwords match, then the 3 appropriate managers are returned as a vector. Which 3 managers are passed back depend on the access level of the user (which can be seen in the user object which was obtained in the database).
Preconditions: None.
Postconditions: The appropriate vector of managers is returned to the caller.

logout (user : User) : void

Description: Releases any locks which may not (for some reason) have been released. The 3 managers each contain a list of all the locks that are taken out for objects of it’s type. All of the locks which belong to the passed in user (the one logging out) are released.
Preconditions: The user was logged in.
Postconditions: All unreleased locks are released.

MiscManager

changePassword (userID : int, newPassword : String) : void

Description: Changes the password of the given User. The String passed in is used as the new password. Updates are reflected in the database.
Preconditions: The userID passed in is valid.
Postconditions: The password for the given User is changed to the String provided and the database is updated.

generateProjectGraph (graphParams : GraphParams) : Graph

Description: Generates a Graph object that contains the information necessary to draw a project graph on the client program’s display. The graph parameters specify which Project to use to create the graph.
Preconditions: None.
Postconditions: A Graph object is returned.

generateConsultantGraph (graphParams : GraphParams) : Graph

Description: Generates a Graph object that contains the information necessary to draw a consultant graph on the client program's display. The graph parameters specify which Consultants to show on the graph.
Preconditions: None.
Postconditions: A Graph object is returned.

generateMachineGraph (graphParams : GraphParams) : Graph

Description: Generates a Graph object that contains the information necessary to draw a machine graph on the client program's display. The graph parameters specify which Machines to show on the graph.
Preconditions: None.
Postconditions: A Graph object is returned.

generateRoomGraph (graphParams : GraphParams) : Graph

Description: Generates a Graph object that contains the information necessary to draw a room graph on the client program's display. The graph parameters specify which Rooms to show on the graph.
Preconditions: None.
Postconditions: A Graph object is returned.

canAddUser () : boolean

Description: Returns true if the user has access to add Users. Otherwise returns false. In the base class (UserManager) true is always returned.
Preconditions: None.
Postconditions: True is returned.

canModifyUser () : boolean

Description: Returns true if the user has access to modify Users. Otherwise returns false. In the base class (UserManager) true is always returned.
Preconditions: None.
Postconditions: True is returned.

canDeleteUser () : boolean

Description: Returns true if the user has access to delete Users. Otherwise returns false. In the base class (UserManager) true is always returned.
Preconditions: None.
Postconditions: True is returned.

getUserList () : Collection

Description: Returns a collection containing the name and ID of all the Users who can log into the system.
Preconditions: None.
Postconditions: A collection containing the names and ID numbers of all Users is returned.

getUser (userID : int) : User

Description: Returns a reference to a user with a given ID.
Preconditions: The userID supplied is valid.
Postconditions: A reference to the User specified is returned.

addUser (newUser : User) : void

Description: Adds the given User to the database. Calls canAddUser() in order to determine if an exception should be thrown or not.
Preconditions: None.
Postconditions: The User passed in is added to the database.

modifyUser (userID : int, user : User) : void

Description: Adds the given User to the database. Calls canModifyUser() in order to determine if an exception should be thrown or not.
Preconditions: None.
Postconditions: The User passed in is added to the database.

deleteUser (userID : int) : void

Description: Adds the given User to the database. Calls canDeleteUser() in order to determine if an exception should be thrown or not.
Preconditions: None.
Postconditions: The User passed in is deleted from the database.

ProjectLeadMiscManager

canAddUser () : boolean

Description: Returns false because ProjectLead Users cannot add users.
Preconditions: None.
Postconditions: False is returned.

canModifyUser () : boolean

Description: Returns false because ProjectLead Users cannot modify users.
Preconditions: None.
Postconditions: False is returned.

canDeleteUser () : boolean

Description: Returns false because ProjectLead Users cannot delete users.
Preconditions: None.
Postconditions: False is returned.

ViewOnlyMiscManager

canAddUser () : boolean

Description: Returns false because ProjectLead Users cannot add users.
Preconditions: None.
Postconditions: False is returned.

canModifyUser () : boolean

Description: Returns false because ProjectLead Users cannot modify users.
Preconditions: None.
Postconditions: False is returned.

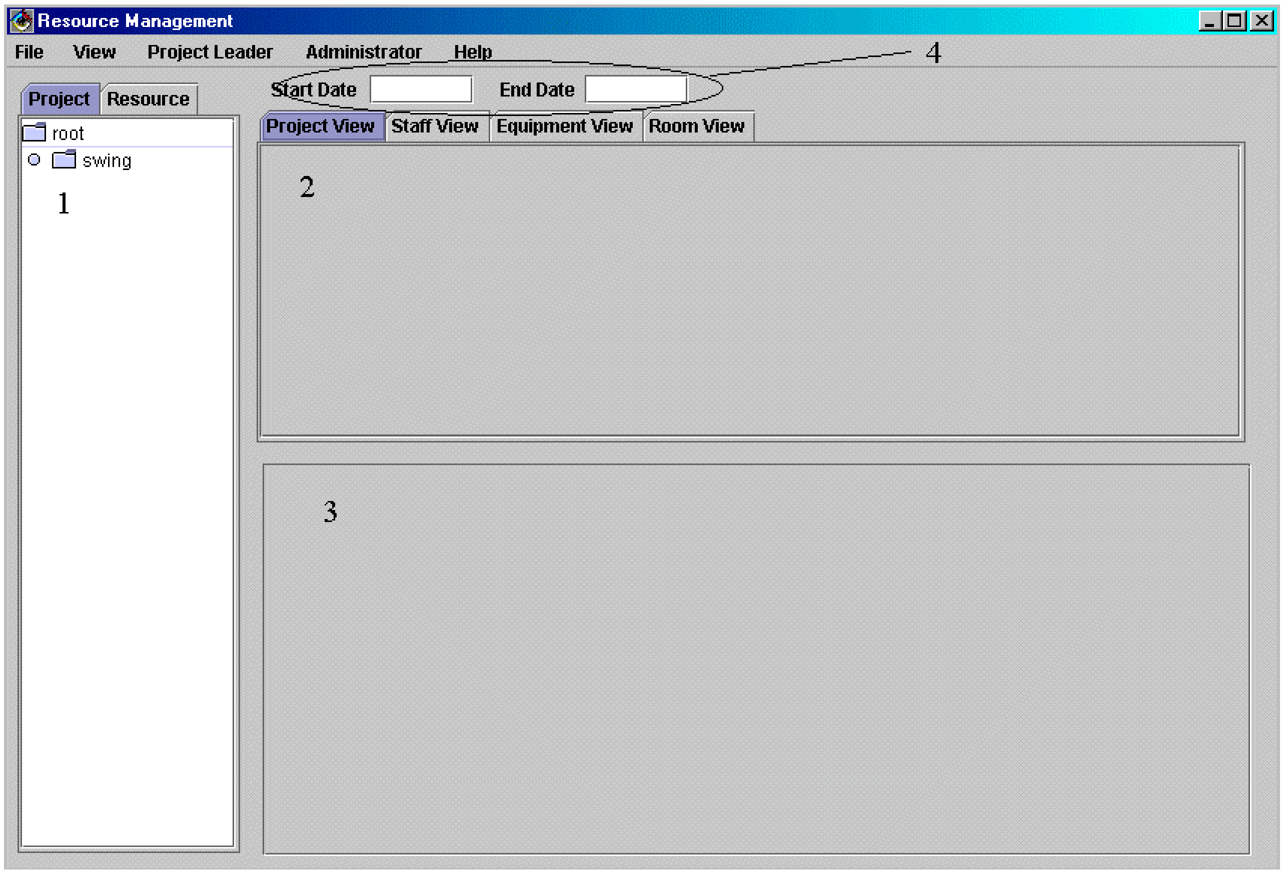
canDeleteUser () : boolean

Description: Returns false because ProjectLead Users cannot delete users.
Preconditions: None.
Postconditions: False is returned.

Graphical User Interface Design

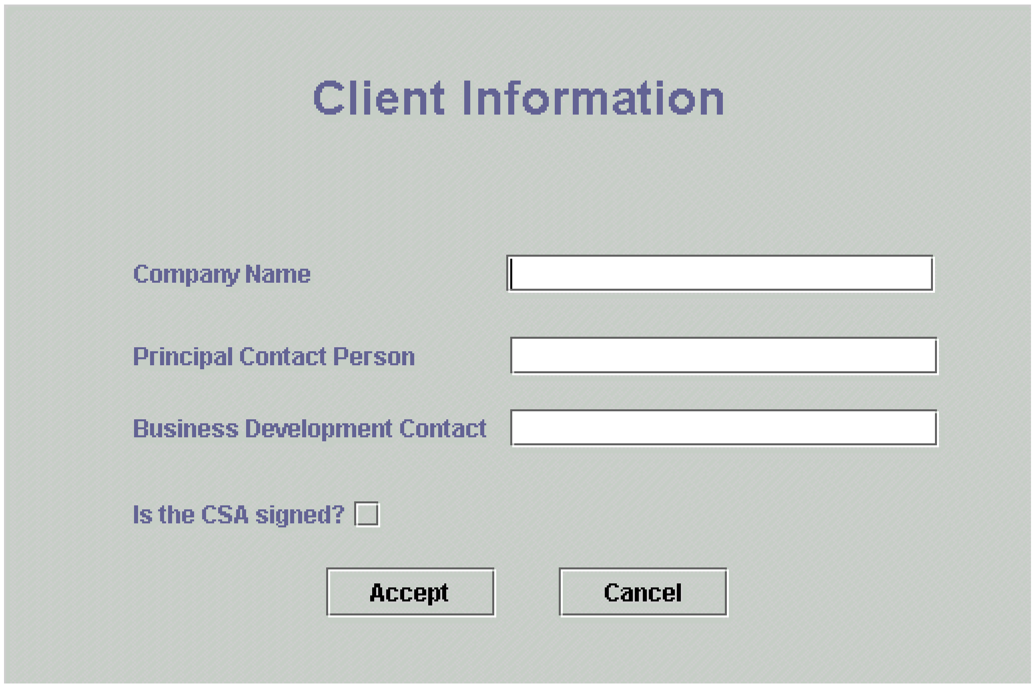
The graphical user interface design consists of two parts. Note that only the layout of the windows, dialog boxes, and graphs has been completed. The underlying logic and event handlers will be developed in the implementation phase.

Windows and Dialog Boxes



The above screenshot is the main interface of the Resource Allocation program. The panel labeled 1 is a tabbed pane. This pane will hold the tree view for both the projects and the resources. The project tree view will contain 4 levels. It will start with the client name which can expand to list the projects, which can expand to list the resource allocations. The fourth level will be the timeslots. Whenever a user selects a particular part of the tree the corresponding form will appear in the info panel which is panel 3. The other tab for panel 1 is the resource tab. When this is selected a tree with two levels will be displayed. The first level will be the resource type and under this will be all the resources of that particular type. When a user selects a particular resource the information regarding that resource will be displayed in panel 3. Panel 2 is a tabbed pane, which will display the corresponding graph depending on the tab selected. The project tab will produce a graph of one project listing all the resources allocated and showing the time allocated. This graph is dependent on the last project that was selected in the tree. All the other tabs generate the particular resource graph and are not dependent upon the selected tree item. The text fields highlighted by number 4 in the above diagram are used in the creation of the graphs. The user will be able to enter the start and end dates for the graph. These dates are used in the generation of the graph and will not normally both be seen in the window at the same time. The user will be able to scroll through the graph to view it from the specified start and end date. The menu bar has the basic menu components, which can be accessed by all users, and special menu components, which can only be accessed by certain user groups. The basic components in the menu bar are the File menu and the View menu. In the file menu the user finds items that will allow the changing of the password and closing of the program. The view menu has an item that will allow the user to see the graph full screen. The special menu components are the project leader menu and the administrator menu. The project leader menu is to be used by the project leaders to add, modify, and delete time allocations. The administrator menu is to be used by the administrator. The administrator will have access to all capabilities of the program. The administrator's menu gives the user the capability to add a client, project, resource, office, resource allocation, and user. It also allows the administrator to view and modify any user.

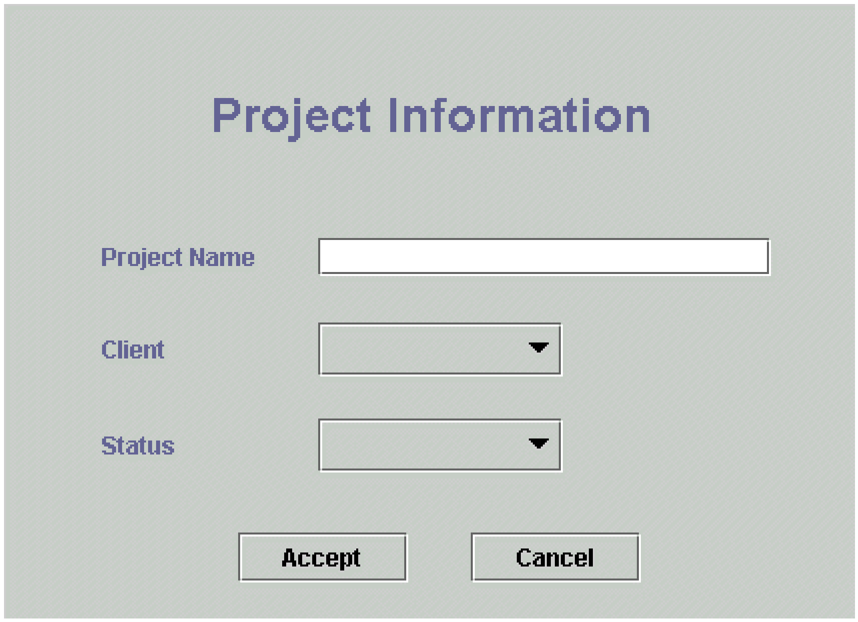
The following screen shots are those of the panels that will be placed in the main user interface in area 3. These panels allow for the information to be neatly displayed as well as to provide a form for the information to be modified. With all of the following panels the buttons at the bottom will only be visible if the user has the capabilities to modify the particular form.


Client Information

Company Name	
Principal Contact Person	
Business Development Contact	

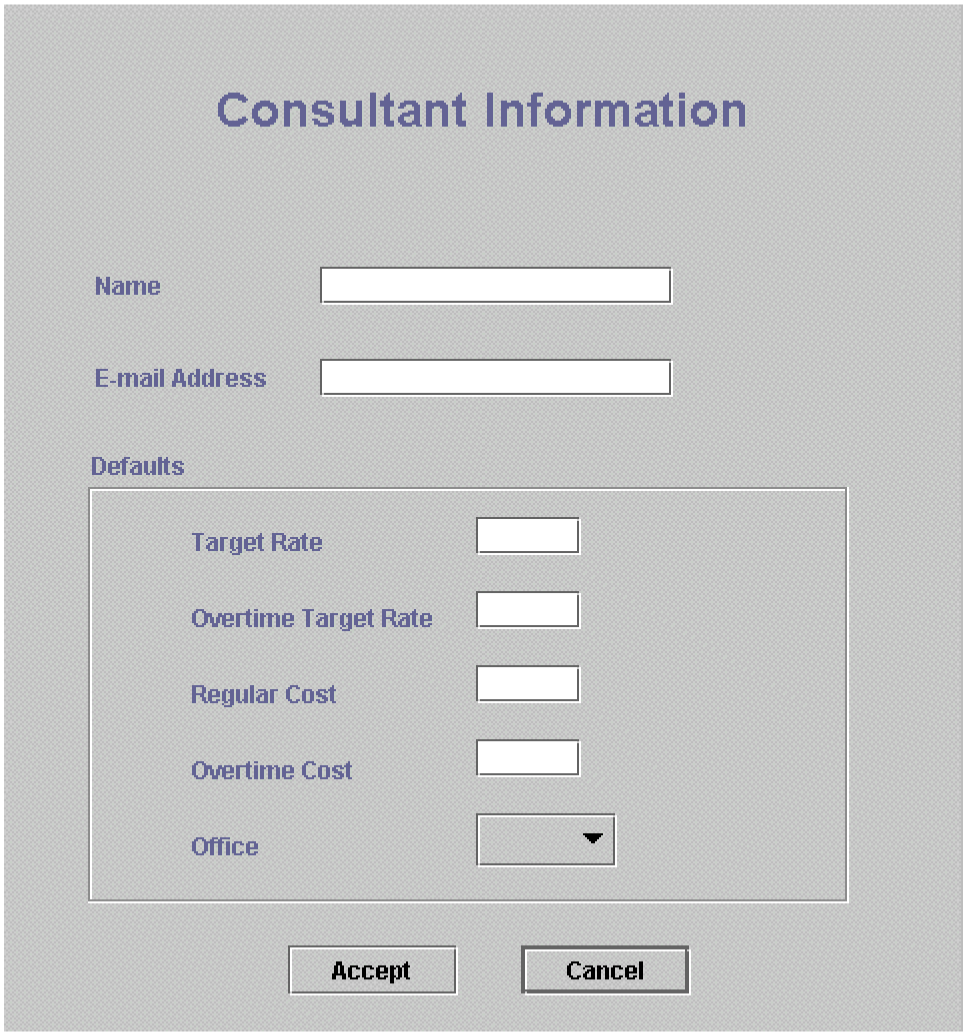
Is the CSA signed? ☐

The above screenshot is that of the client information. It will display the company's name as well as contact information. It also provides a checkbox indicating whether or not the CSA, consulting services agreement, is signed.



The above screenshot would be displayed in panel 3 of the main GUI when any project is selected and will fill in the fields according the project. The project name is the name of the project.   The client is who the project is being worked on for. When this is being modified or created the client must be chosen from a list of clients that has been already entered into the system. The status is the status of the project. There are 4 status levels: discovery, proposal, allocated, and terminated.

The next three screenshots are of the different resource panels. Each resource has common information but the resources also have information that is specific to that type which resulted in the creation of a separate panel for each type.

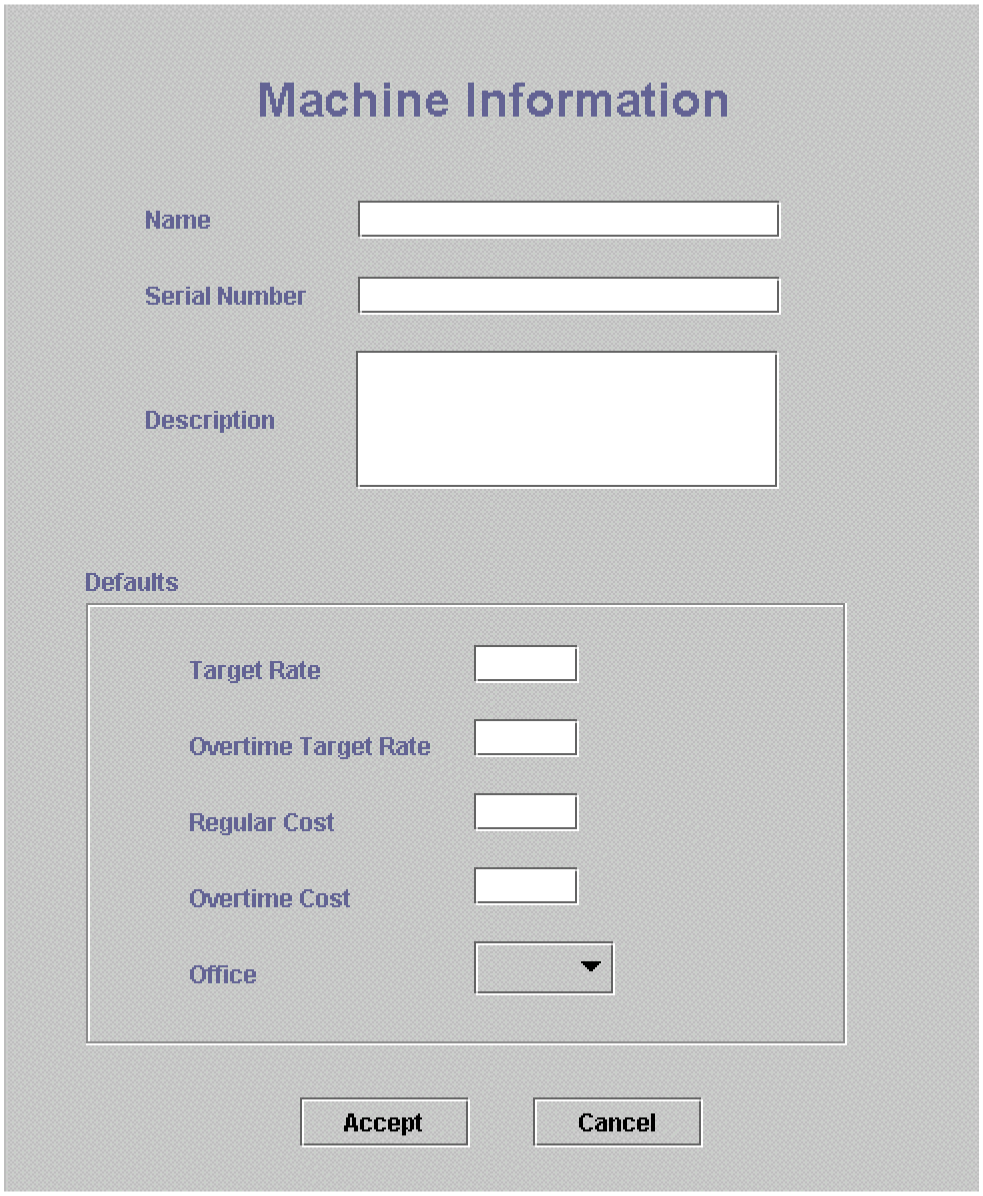

Consultant Information

Name	
E-mail Address	

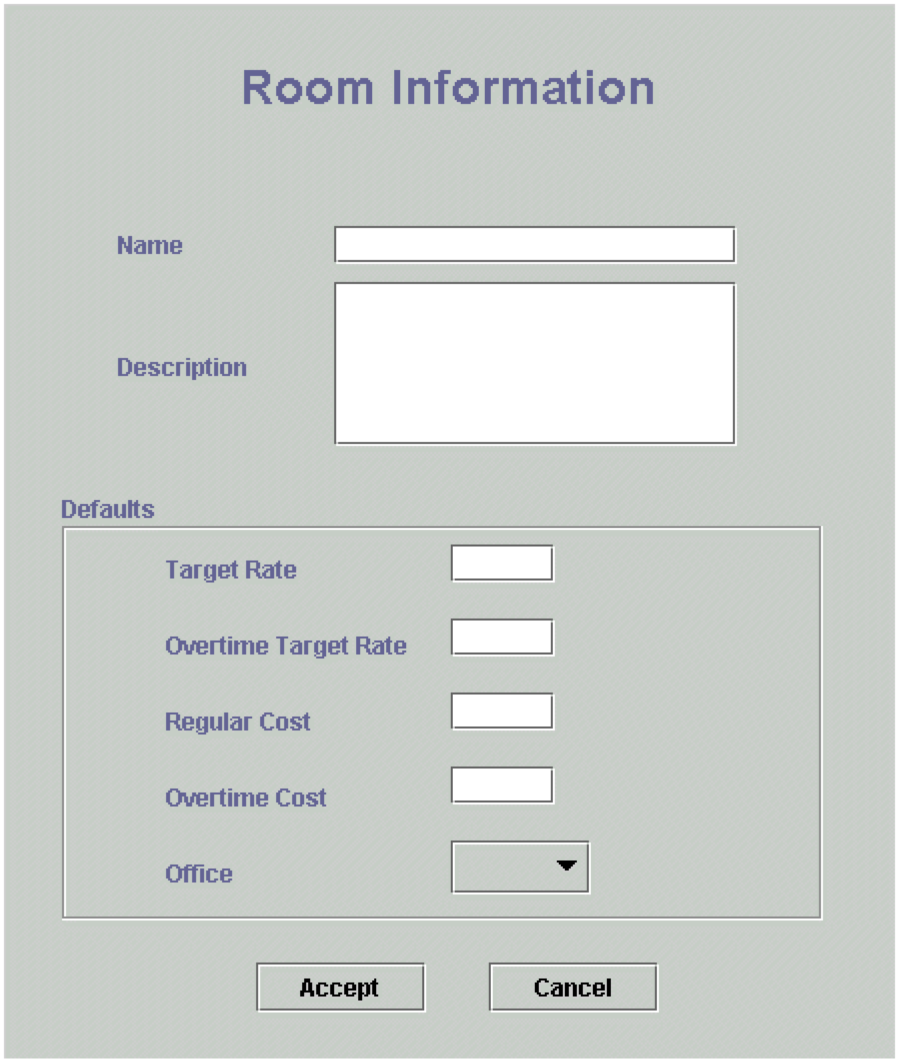
Defaults

Target Rate	
Overtime Target Rate	
Regular Cost	
Overtime Cost	
Office	

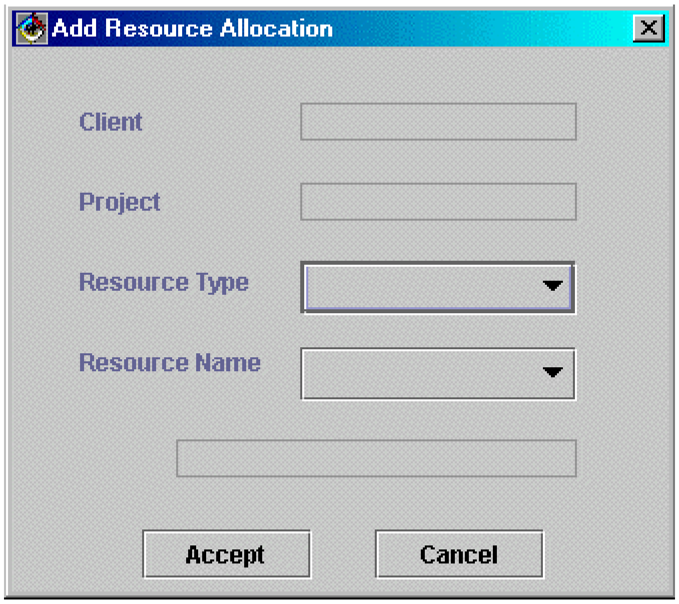
The above figure corresponds to the screenshot of the consultant information panel. The information that is specific to the consultant is the name and the email address. The other information is common for all resources, although the exact values are dependent upon the particular item. The target rate is the desired rate that the consultant should be allocated for. The overtime rate for the consultant when allocated. The regular and overtime costs are what the particular resource costs. These are used as the recommended minimum values in the timeslot portion. The last item in the default area is the office, which is the location that the resource is being used at.



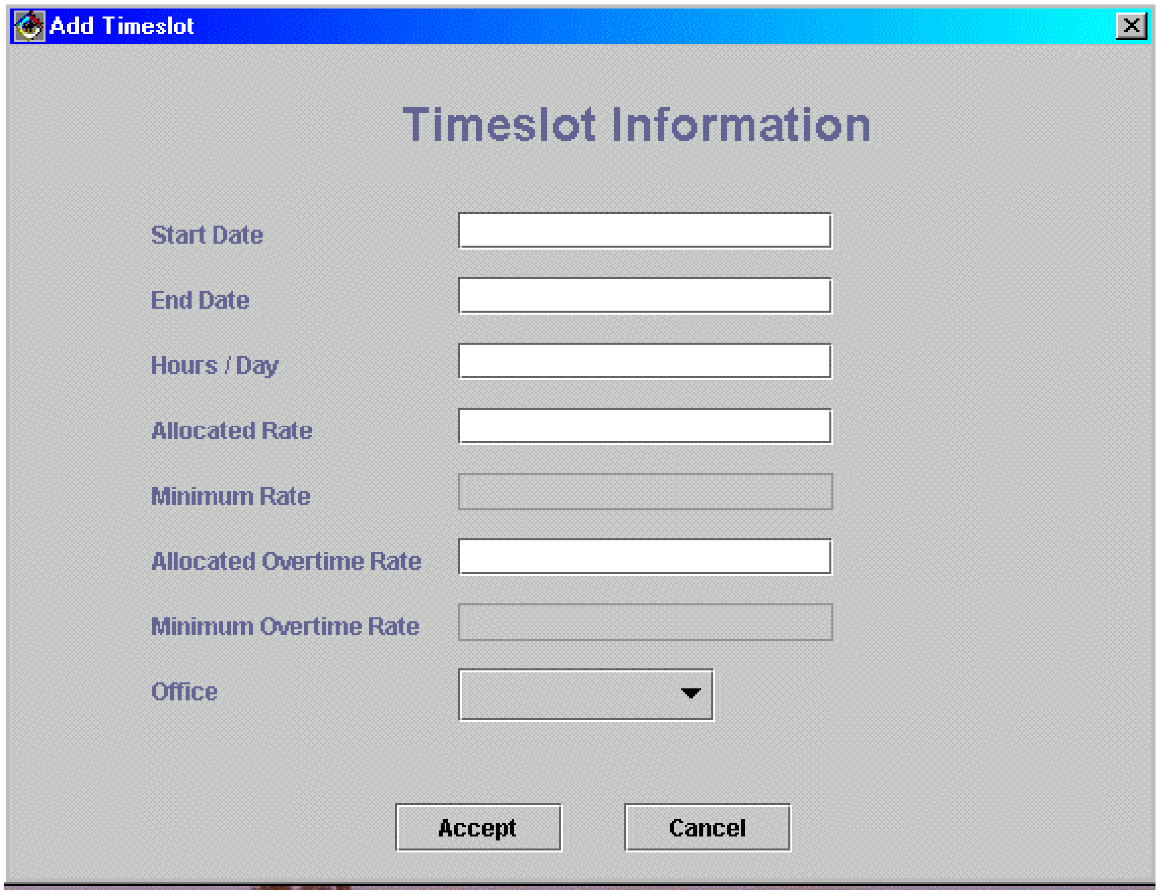
The above figure shows the screenshot for the panel that is activated when a resource of type machine is chosen. It again has the same default information as the other resource types. The information that is specific to this type is the name, serial number and description. The name is the name of the machine, and the serial number is its serial number. The description allows the user to enter a description in order to tell what the particular machine is to be used for.



The above screenshot is the last of the resource panels and is used for the resources of type room. The room resource has the common information as well as a name and a description, which is very similar to the description for a machine.

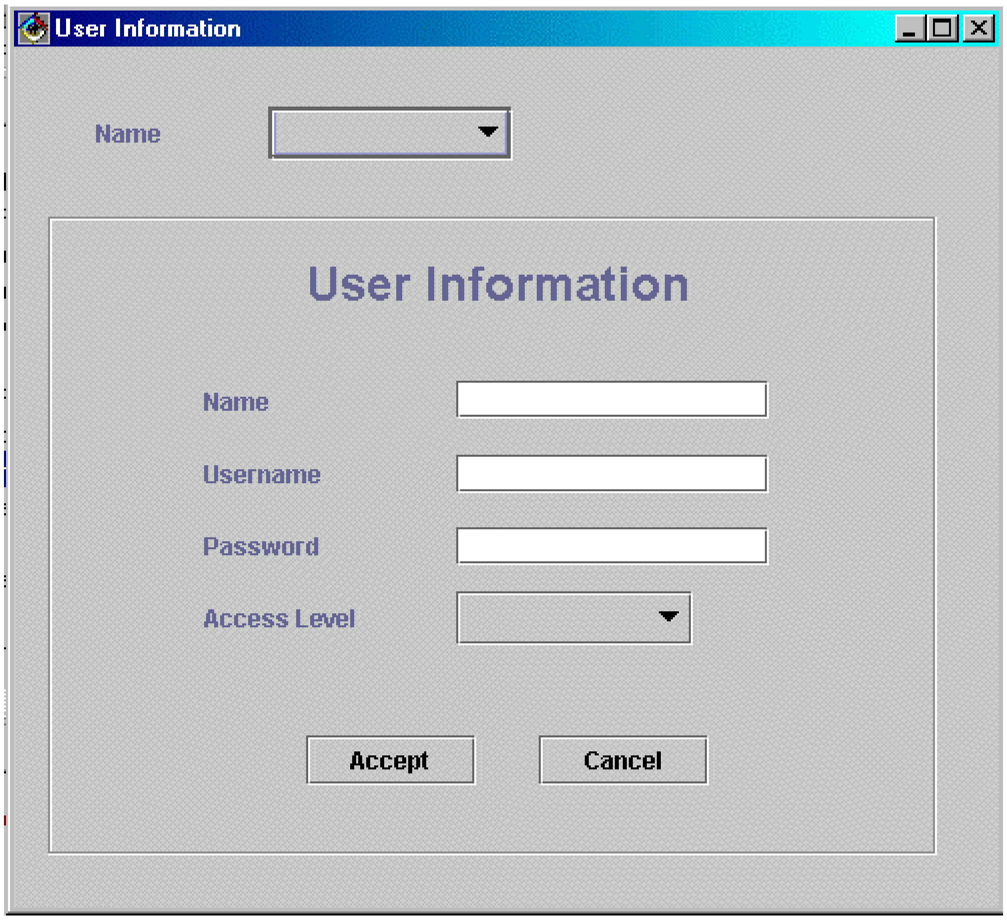


The above dialog box is displayed when the user requests that a ResourceAllocation be added to a Project. The Client and Project are shown in non-editable text boxes. The user selects a resource type from the combo box. A list of resources of that type are displayed in the Resource Name combo box. The user can select the particular resource to schedule from this box. The non-editable text box below the Resource Name combo box contains information that further identifies the resource selected in case of duplicate resource names. For machines, this box will contain the serial number; for consultants, it will contain the email address; for rooms, it will contain the room description.

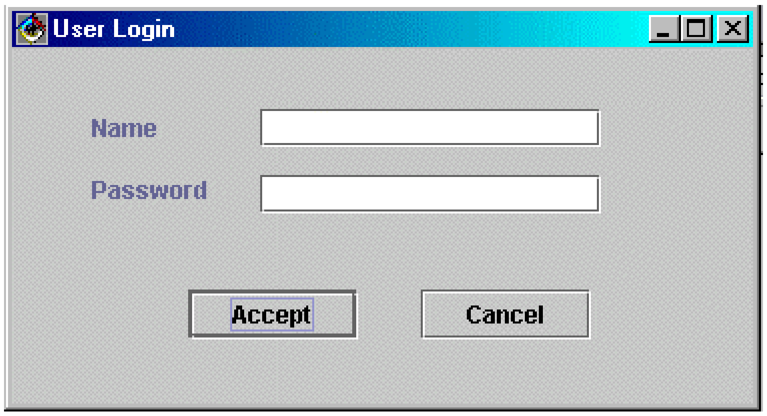


The above panel is displayed when the user selects a timeslot for a particular resource allocation. The start and end dates mark the period of time that the resource is being used on the project. The hours/day are the amount of hours that the particular resource will be used on the project. The allocated rate and allocated overtime rate are rates at which the particular resource is allocated for in regards to this particular time period. The minimum rate and minimum overtime rate are advisory and are determined by the default cost of the resource.

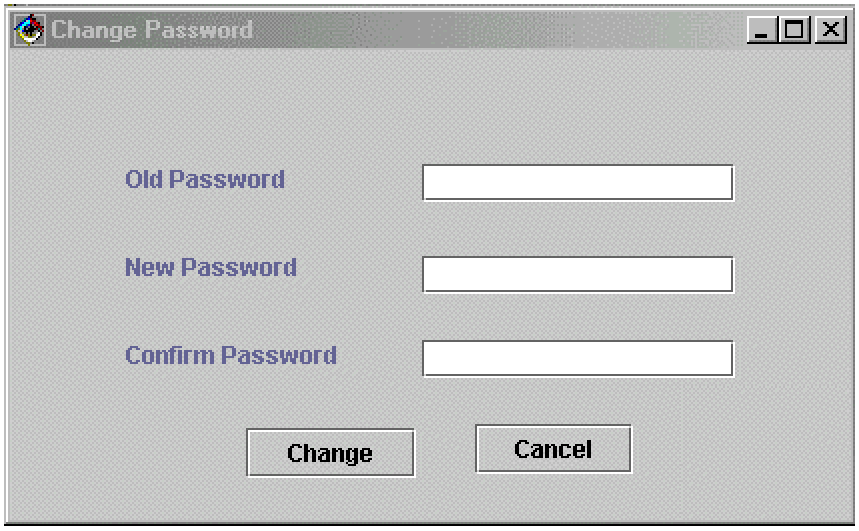
When a user chooses to add a new client, project, or resource the same panels will appear in a separate dialog box. The user is then able to enter the needed information to create the new item to be used in scheduling.



The above screenshot shows the information on a particular user as well as allows changes to be made to this information. The administrator selects a name from the choice box, which will be filled, with the names of all the users of the system. When the administrator has selected a name the information regarding the selected user will appear in the lower panel. The name, username and access level will be visible to the administrator but the administrator will not be able to see the user's password. The password field is used when the administrator needs to reset the user's password. The bottom portion of the dialog is used when a new user is added.



The very first GUI element that the user will see is the login dialog box. This dialog (shown above) allows the user to enter their username and password to enter the system.



The change password dialog is similar to that of any program and takes in the users old password, the new password and a confirmation of the new password in order to verify that there were no typos. The GUI uses the Java password fields, which automatically do not display what the user types into them.

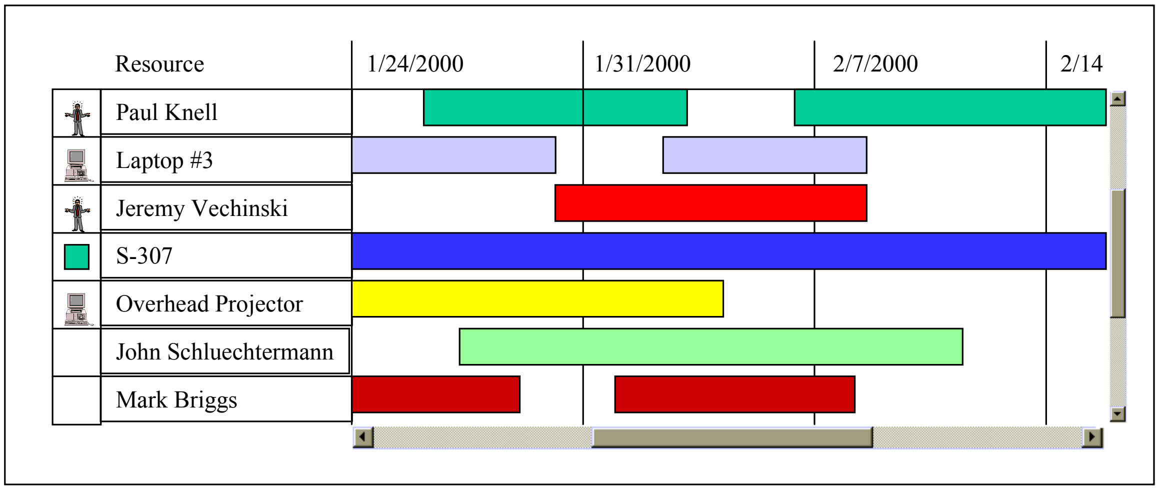
Project and Resource Graphs

Common features of the graphs

The graphs are split up into logical sub-components called blocks. Each block contains resource icon, resource name, and a vector of rectangles. This way, each resource and it's associated data are kept together.

The major computation for the graphs (e.g. where the rectangles are to be drawn in the viewport) is done on the server. Then, only these points are passed back over the network to the client and displayed in the viewport.

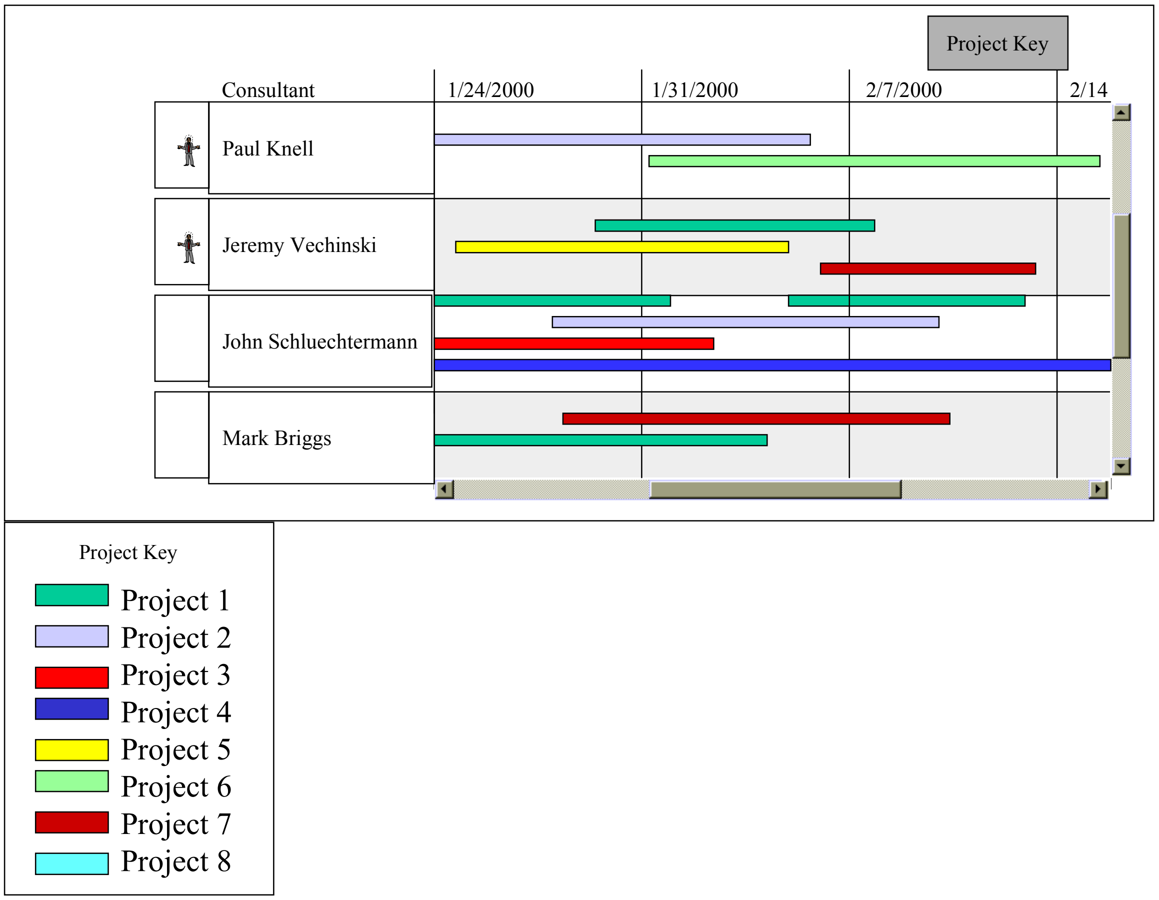
Project View Graph



The project view graph is made up of a single JScrollPane. Our scroll pane, however, is made up of 4 different sections: a viewport, a column header, a row header, and a corner component. The viewport is where the rectangles, for the graph, are drawn. The column header contains the dates along the top of the graph. The row header contains the names of the resources. The corner component simply contains the string "Resource." The rectangles are different colors to enhance clarity and there is always only one rectangle per resource.

The column header, row header, and corner component are made so that the resource names and the dates don't scroll out of the pane when the graph is scrolled.

Resource View Graph
(basically same for "Staffing View," "Machine View," and "Room View")



The resource view graphs are made up of a single JScrollPane. Our scroll pane, however, is made up of 4 different sections: a viewport, a column header, a row header, and a corner component. The viewport is where the rectangles, for the graph, are drawn. The column header contains the dates along the top of the graph. The row header contains the names of the resources. The corner component simply contains a string (either "Consultant," "Machine," or "Room." The rectangles are different colors to enhance clarity. There may be more than one rectangle per resource – the number of rectangles equals the number of projects that the resource is on during the specified start and stop date. The resource graph also contains a "project key" and a button to show the project key.

The column header, row header, and corner component are made so that the resource names and the dates don't scroll out of the pane when the graph is scrolled.

Conclusions

This document provides all the information related to the current design of the Resource Management Software system. By following the design procedure outlined in the Larman book the team was able to develop an implementation plan for a relatively complicated software system in about five weeks. The details contained in the sequence diagrams, class diagrams, and system contracts will be invaluable when it comes time to code the software.

While the majority of the system has been designed, there are still a few areas lacking the detail required for implementation. The specifics regarding the use of the object database have not been worked out. Also, event handling in the GUI has not been discussed in detail. Algorithms for generation of the graphs have been suggested, but they haven't been finalized. Some frill features don't appear in the current design. The design of features like the scheduling conflict report and revenue forecasting will be deferred until the core system is implemented. Since they are not especially important to our client, they will only be completed if time allows.

Although a few unresolved issues exist, it is believed that the core of the system has been completely designed. The critical objects in the Entity and Controller packages should change little if it all during the rest of development.

Work will now shift to implementation. Team members will split up and be responsible for implementing one of the packages. Individuals will have to complete any low level sequence diagrams that pertain to operations within the package. Team members may also have to fill in any details they find missing in the design. The design information created focused on defining the interfaces between the packages. The hope is that while some of the implementation details within the packages will vary from the current design, none of the interfaces will have to be drastically changed.

Implementation will take place in three phases. The goal of each phase is to have a working software system that has a certain set of features. The purpose of having multiple phases is to gradually add functionality to a core program. The first implementation phase will involve coding all the entity classes. In parallel with this coding effort, work will continue on resolving some of the outstanding issues mentioned above. At the end of first phase, the goal is to have a system that allows users to log in and display information contained in the database. The second implementation phase will primarily involve coding the controller classes and expanding the GUI. At the end of phase two, the goal is to have a software system where the user can add and modify information in the database and display resource and project graphs. Finally, phase three will focus on adding frill features. At the end of phase three, the goal is to have a software system that provides user management, revenue forecasting, and a variety of reports.

Works Cited

Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall, 1998.

Appendix A – Specifications Document

I. **Client**

a. Each Client has several attributes:
   - Name – The name of the client company.
   - signedCSA – a boolean value that is true only if the client has signed the Consulting Services Agreement.
   - Principal Name – The name of the managing Principal of the client organization.
   - Business Development Name – The name of the business development person at the client organization.

b. Clients are only created and modified by Admin-level users.

c. Clients can only be deleted if there are no Projects assigned to them.

II. **Project**

a. Each project has a status attribute.
   1. This attribute can have one of four possible values:
      - 0 = Project is in the “Discovery” phase.
      - 1 = Project is in the “Proposal” phase.
      - 2 = Project is in the “Allocated” phase.
      - 3 = Project is in the “Terminated” phase.

      The project can only progress forward from Discovery to Allocated (and not backwards), however it can go to Terminated at any time.

b. Each project has a name attribute.
   1. The name is a string and must be unique among all other projects within a particular client. The user will always select the client first before selecting a project.
   2. A project has zero or more resources allocated to it. See Section IV on Resource Allocations.

c. A project has exactly one specified Client. In the selection tree, the project will be listed as a child of the specified client.

d. A project can be deleted. However, it is only fully removed from the system if it has no Resource Allocations. If it has Resource Allocations, then instead of removing the project from the system it will be put into the “Terminated” state. Please refer to item ‘a’ of this section. As a terminated project, the information is still in the database and is still viewable. However, the terminated project may not be modified in any way. Currently, there is no spec. for un-terminating a project. (This might be a future enhancement.) Resource Allocations within a terminated project may be not modified or deleted.

III. **Resource**

a. Three types of resources
   1. Consultant – In addition to the features common to all resources, each Consultant will have an attribute for e-mail address.
   2. Machine – In addition to the features common to all resources, each Machine will have an attribute for it’s serial number.
   3. Room – In addition to the features common to all resources, each Room will have a description attribute.

b. Common features of resources

1. Office. Resources such as Consultant and Machine are portable. However, they still have a "default office" which is the typical office in which they work/exist. Resources such as Rooms also have an office, which is the office that physically contains them. When creating a Timeslot within a Resource Allocation for this portable resource, the user will be presented with a form that allows them to specify the location. The default value of this field will be the default office. When creating a Timeslot within a Resource Allocation for this Room, the user will be presented a similar form, and the room's office will be shown but would not be modifiable.
2. Target Rates. The target rate is the default per-hour-rate that the Client will be charged for the resource. This default will be modifiable similar to how a portable resource's office can be modified in the form for creating a Timeslot within some resource allocation for this resource. In this case, the behavior is similar to Office (part 1 of this section) however there is no distinction between portable and non-portable resources. When creating a resource, the target rate is modifiable but the default that initially appears in the "Add Resource" form will be generated using a known formula.
3. Costs. Each resource has a cost and an overtime cost which represents the per-hour-cost incurred on the consulting company for the resource being used. When a Timeslot is added or modified the cost is displayed as a label called "Minimum Rate" and "Minimum Overtime Rate". If the rate (or overtime rate) assigned to the Timeslot is below the minimum (the cost) a warning is issued to the user.

c. A resource may be deleted only if it is not allocated to any projects (terminated or not).

IV. Resource Allocations

a. Resources can be added to projects, this is called a Resource Allocation. The allocation can consist of zero or more Timeslots, where each represents a block of time for which the resource is allocated to the project. A project in the "Discovery" phase cannot have any resources allocated to it. A project in the "Proposal" phase can have resources allocated (with Timeslots), but these allocations are not used in revenue forecasting. See section VII for more information on revenue forecasting.

b. Conflict checking will be performed when Timeslots are added to a Resource Allocation or when they are modified. Conflict checking insures that the resource is not scheduled to be in multiple offices at once and that the resource is not scheduled for a time when it is already scheduled. If either of these is the case, then an appropriate warning message will be displayed to the user.

c. If a Timeslot is added or modified such that it will overlap other Timeslot(s) within the same project and resource (a conflict within the same project), then the user will be prompted as to whether he/she really wants to continue with this action. If the user chooses to continue, the new Timeslot will be merged with all overlapping pairs (in the same project) and the net result will be a single Timeslot where the begin time is the earliest begin time of any of the involved pairs and the end time is the latest time of any involved pairs. The pairs involved in this "merge" are only the pairs that have the same office location as the new pair we are trying to add.

d. If a ResourceAllocation gets deleted, then all Timeslots within it get deleted and no record is kept. There is no "terminated" type of state for a ResourceAllocation.

V. Users

a. Each user has a login name and password.

b. Each user has a full name in addition to the unique login name.
c. Each user has an access level, which can have one of three values:
   - 0 = ViewOnly Level. The user cannot modify any information and can only view certain non-confidential information. The user can change his/her password.
   - 1 = ProjectLead Level. The user may modify the Resource Allocations of any project. (In the future, the user might be assigned to certain projects and only able to modify within assigned projects. However, the current specs. do not require this feature.) The user can view any non-confidential information. The user can change his/her password.
   - 2 = Admin Level. The user may add/modify/delete Clients, Projects, Resources, Resource Allocations, and Users. The user can view all information in the system. The user may not delete himself from the system, but can add/modify/delete other users including modification of usernames, passwords, and access levels.

VI. Proposed Project Schedule (PPS)

a. When a project is created, it starts in the "Discovery" state. In this state, PPS is displayed in the "Project View Graph", and nothing else (because there are not resource allocations yet.)
b. The PPS consists of a start date and end date and these two pieces of data are specified when the project is created. They can be changed when the project is modified.
c. When a project is in the proposal state, the PPS can be modified, and gets displayed in the "Project View" along with all the resource allocations.
d. When a project is in the allocated state, the PPS could still be modified but the user wouldn't ever do this because it is not displayed in the "Project View". Instead, the "merge" of all the resources allocations is displayed instead.
e. The PPS is automatically extended if a Timeslot is added to a resource allocation of that project which exceeds the bounds of the PPS.
f. The PPS is always a single block of time.

VII. Server Application

a. There will be a single server, which is a Java application that uses JDK 1.2, that interfaces with the PSE Pro database and communicates with all GUIs using remote method invocation (RMI). This server will allow the system to be used by many users at once.
b. When the server starts, it will open the database and load all the User objects into RAM. A container obtain will be used to contain these User objects. A LoginManager object will be created and given a reference to the container of User objects. The LoginManager will be a remote object that is directly accessible from the RMI Naming Service. The LoginManager will be capable of authorizing a login and will return objects (if the login was successful) that allow the GUI to interface to the rest of the system. This method will resolve security issues.

VIII. The Graphical User Interface (GUI)

a. See separate screen shots

IX. Text Reports

a. Text reports will be generated when the user selects the appropriate menu option. The results will be displayed in a window, and sent to a file on the local hard-drive.

When a report is requested, a time interval is specified. One possibility for implementation is to use a Java Servlet to “serve” the information to the client in HTML. Another possibility is to stream the HTML text to the client and have the GUI start up a web-browser that points at the local URL.

b. One type of text report will be the “Staffing Report”. This report will have a section for each consultant. It will list the projects that the consultant has worked on during the specified time interval. It will show the consultant details such as name, e-mail, rate/cost, etc. For each project listed, it will show the start date and end date, the total number of hours that the consultant is scheduled to work for that project, and the forecasted revenue for that consultant and that project.
c. Another type of text report will be the “Conflict Report” This report will list each resource, and all the conflicts in the system associated with that resource.

X. Graphs

a. Project View
b. Resource View – The resource view consists of three graphs. Each will be located on a different tab within the GUI.
    1. Consultant View
    2. Machine View
    3. Room View

XI. Revenue Forecasting

a. The user will be able to enter a start date (which defaults to the current date) and an end date. The program will be able to generate revenue, cost, and profit over this time interval for either a selected project or all projects. The report will be displayed in a window and saved to a text file.
b. The revenue calculation will be done by calculating the total number of hours each week allocated in a particular resource allocation. If that number exceeds 40 hours (or some other threshold value specified in a properties file) the excess hours will be counted as overtime hours. The profit for that week is the weighted average of the regular rates associated with the Timeslots for that week (for the first 40 hours) plus the weighted average of the overtime rates for the remaining hours.
c. The cost calculation will be done by multiplying the total number of work hours available (regardless of whether the resource was fully used) for the time period specified. The cost associated with each resource will be multiplied by this value (the overtime cost will be used for all billable hours beyond 40). The sum of these costs will be found to get the net cost.
d. Profit will be calculated by subtracting cost from the all-projects revenue calculation.

Appendix B – Glossary

Actual Rate – The per hour rate that a client is actually charged when a resource is used on a project. This rate can change from project to project.

Client – A company that requests the services of ABC Corporation.

Consultant – One of the three resource types. It is any human that works on a project.

Consulting Services Agreement (CSA) – A contract that exists between a client and ABC Corporation.

Cost – The cost of the resource per hour of use. It is the amount of money ABC Corporation must pay for using the resource.

Default Office – The office where a resource object is primarily used.

Machine – One of the three resource types. It is any inanimate object used to facilitate the completion of the project. Examples of machines include computers and projectors.

Minimum Rate – The minimum hourly rate that should be charged for a particular resource. It is equal to the cost of the resource. Charging a client the minimum rate would mean ABC Corporation would break even.

Office – A location where the ABC Corporation has an office. Examples of offices include "Milwaukee office," "Minneapolis office," and "Chicago office."

Project – A task that is to be performed by ABC Corporation. Clients request that projects be performed.

Proposed Project Schedule (PPS) – Consists of a start and end date specified when a project is created. Gives an overall time frame in which a project will be performed.

Resource – An object used to facilitate the completion of a project. Resources are one of three types: room, equipment, or consultant.

Resource Allocation – The assignment of a resource (consultant, equipment, or room) to a particular project. The resource allocation just makes the resource available for use on a project. It does not "schedule" a resource for a particular time frame.

Room – One of the three resource types. It represents a single room in a particular office.

Timeslot – A time frame in which a resource is scheduled to work on a particular project. A timeslot specifies the start and end date of a resource commitment. A resource must be allocated to a project before any timeslots can be added. Any number timeslots can be associated with a single resource allocation.

Appendix C – Java Code for the GUI

AboutDialog.java

```
//--------------------------------------------------------------------------
// AboutDialog.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//--------------------------------------------------------------------------


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

/**
 * A dialog class that is used with the addition of a client.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the consultant.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AboutDialog extends  JDialog {
// Private data members
       private  JLabel ivjheadingLabel = null;
       private  JPanel ivjJDialogContentPane = null;
       private  JLabel ivjjeremyLabel = null;
       private  JLabel ivjjohnLabel = null;
       private  JLabel ivjmarkLabel = null;
       private  JButton ivjokButton = null;
       private  JLabel ivjpaulLabel = null;

/**
 * AboutDialog constructor comment.
 */
public AboutDialog() {
       super();
       initialize();
}

/**
 * AboutDialog constructor comment.
 * @param owner  Frame
 */
public AboutDialog( Frame owner) {
       super(owner);
}

/**
 * AboutDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AboutDialog( Frame owner, String title) {
	super(owner, title);
}

/**
 * AboutDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AboutDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}

/**
 * AboutDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AboutDialog( Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the headingLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getheadingLabel() {
	if (ivjheadingLabel == null) {
		try {
			ivjheadingLabel = new  JLabel();
			ivjheadingLabel.setName("headingLabel");
			ivjheadingLabel.setText("Software Written By:");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjheadingLabel;
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(new  GridBagLayout());

			GridBagConstraints constraintsheadingLabel = new  GridBagConstraints();
			constraintsheadingLabel.gridx = 1; constraintsheadingLabel.gridy = 1;
			constraintsheadingLabel.ipadx = 24;
			constraintsheadingLabel.insets = new  Insets(20, 37, 10, 248);
			getJDialogContentPane().add(getheadingLabel(), constraintsheadingLabel);

			GridBagConstraints constraintsmarkLabel = new  GridBagConstraints();
			constraintsmarkLabel.gridx = 1; constraintsmarkLabel.gridy = 2;
			constraintsmarkLabel.ipadx = 40;
			constraintsmarkLabel.insets = new  Insets(10, 99, 10, 218);
			getJDialogContentPane().add(getmarkLabel(), constraintsmarkLabel);

			GridBagConstraints constraintspaulLabel = new  GridBagConstraints();
			constraintspaulLabel.gridx = 1; constraintspaulLabel.gridy = 3;
			constraintspaulLabel.ipadx = 27;
			constraintspaulLabel.insets = new  Insets(10, 99, 10, 244);
			getJDialogContentPane().add(getpaulLabel(), constraintspaulLabel);

			GridBagConstraints constraintsjeremyLabel = new  GridBagConstraints();
			constraintsjeremyLabel.gridx = 1; constraintsjeremyLabel.gridy = 5;
			constraintsjeremyLabel.ipadx = 23;
			constraintsjeremyLabel.insets = new  Insets(10, 99, 10, 202);
			getJDialogContentPane().add(getjeremyLabel(), constraintsjeremyLabel);

			GridBagConstraints constraintsokButton = new  GridBagConstraints();
			constraintsokButton.gridx = 1; constraintsokButton.gridy = 6;
			constraintsokButton.ipadx = 34;
			constraintsokButton.insets = new  Insets(10, 170, 20, 171);
			getJDialogContentPane().add(getokButton(), constraintsokButton);

			GridBagConstraints constraintsjohnLabel = new  GridBagConstraints();
			constraintsjohnLabel.gridx = 1; constraintsjohnLabel.gridy = 4;
			constraintsjohnLabel.ipadx = 17;
			constraintsjohnLabel.insets = new  Insets(10, 99, 10, 178);
			getJDialogContentPane().add(getjohnLabel(), constraintsjohnLabel);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the jeremyLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getjeremyLabel() {
	if (ivjjeremyLabel == null) {
		try {
			ivjjeremyLabel = new  JLabel();
			ivjjeremyLabel.setName("jeremyLabel");
			ivjjeremyLabel.setText("Jeremy Vechinski");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjjeremyLabel;
}

/**
 * Return the johnLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getjohnLabel() {
	if (ivjjohnLabel == null) {
		try {
			ivjjohnLabel = new  JLabel();
			ivjjohnLabel.setName("johnLabel");
			ivjjohnLabel.setText("John Schluechtermann");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjjohnLabel;
}

/**
 * Return the markLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getmarkLabel() {
	if (ivjmarkLabel == null) {
		try {
			ivjmarkLabel = new  JLabel();
			ivjmarkLabel.setName("markLabel");
			ivjmarkLabel.setText("Mark Briggs");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjmarkLabel;
}

/**
 * Return the okButton property value.
 * @return  JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JButton getokButton() {
	if (ivjokButton == null) {
		try {
			ivjokButton = new  JButton();
			ivjokButton.setName("okButton");
			ivjokButton.setText("OK");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjokButton;
}

/**
 * Return the paulLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getpaulLabel() {
	if (ivjpaulLabel == null) {
		try {
			ivjpaulLabel = new  JLabel();
			ivjpaulLabel.setName("paulLabel");
			ivjpaulLabel.setText("Paul Knell");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjpaulLabel;
}

/**
```

```
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AboutDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(426, 240);
		setTitle("About Resource Management");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		AboutDialog aAboutDialog;
		aAboutDialog = new AboutDialog();
		aAboutDialog.setModal(true);
		aAboutDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aAboutDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
	}
}
}

/*
 * Revision History
 *
 */
```

AddClientDialog.java

```
//------------------------------------------------------------------------------
// AddClientDialog.java
//
// Author: Mark Briggs
//               John Schluechtermann
//
// GUI creaetd graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified 1/29/00
```

```
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//------------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the addition of a client.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the consultant.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/
class AddClientDialog extends  JDialog {
// Private data members
private ClientFormPanel ivjClientFormPanel = null;
private  JPanel ivjJDialogContentPane = null;

/**
 * AddClientDialog constructor comment.
 */
public AddClientDialog() {
       super();
       initialize();
}

/**
 * AddClientDialog constructor comment.
 *
 * @param owner  Frame
 */
public AddClientDialog(Frame owner) {
       super(owner);
}

/**
 * AddClientDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddClientDialog(Frame owner, String title) {
       super(owner, title);
}
/**
 * AddClientDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddClientDialog(Frame owner, String title, boolean modal) {
       super(owner, title, modal);
}

/**
 * AddClientDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddClientDialog(Frame owner, boolean modal) {
       super(owner, modal);
}

/**
 * Return the ClientFormPanel property value.
 * @return ClientFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ClientFormPanel getClientFormPanel() {
	if (ivjClientFormPanel == null) {
		try {
			ivjClientFormPanel = new ClientFormPanel();
			ivjClientFormPanel.setName("ClientFormPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjClientFormPanel;
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(new  GridBagLayout());

			GridBagConstraints constraintsClientFormPanel = new  GridBagConstraints();
			constraintsClientFormPanel.gridx = 1; constraintsClientFormPanel.gridy =
			1;
			constraintsClientFormPanel.fill =  GridBagConstraints.BOTH;
			constraintsClientFormPanel.weightx = 1.0;
			constraintsClientFormPanel.weighty = 1.0;
			constraintsClientFormPanel.insets = new  Insets(6, 4, 14, 10);
			getJDialogContentPane().add(getClientFormPanel(),
			constraintsClientFormPanel);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddClientDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(540, 378);
		setTitle("Add Client");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		AddClientDialog aAddClientDialog;
		aAddClientDialog = new AddClientDialog();
		aAddClientDialog.setModal(true);
		aAddClientDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aAddClientDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
	}
}
}

/*
 * Revision History
 *
 */
```

AddConsultantDialog.java

```
//---------------------------------------------------------------------------
// AddConsultantDialog.java
//
// Author: Mark Briggs
//                          John Schluechtermann
//
// GUI creaetd graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the addition of a consultant.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the consultant.
 *
 * @version   1.0
 * @author           Mark Briggs
 * @author            John Schluechtermann
 **/

class AddConsultantDialog extends  JDialog {
       private ConsultantFormPanel ivjConsultantFormPanel = null;
       private  JPanel ivjJDialogContentPane = null;
/**
 * AddConsultantDialog constructor comment.
 */
public AddConsultantDialog() {
       super();
       initialize();
}
/**
 * AddConsultantDialog constructor comment.
 * @param owner  Frame
 */
public AddConsultantDialog( Frame owner) {
       super(owner);
}
/**
 * AddConsultantDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddConsultantDialog( Frame owner, String title) {
       super(owner, title);
}
/**
 * AddConsultantDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddConsultantDialog( Frame owner, String title, boolean modal) {
       super(owner, title, modal);
}
/**
 * AddConsultantDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddConsultantDialog( Frame owner, boolean modal) {
       super(owner, modal);
}
/**
 * Return the ConsultantFormPanel property value.
 * @return ConsultantFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ConsultantFormPanel getConsultantFormPanel() {
       if (ivjConsultantFormPanel == null) {
              try {
                     ivjConsultantFormPanel = new ConsultantFormPanel();
                     ivjConsultantFormPanel.setName("ConsultantFormPanel");
                     ivjConsultantFormPanel.setBounds(7, 7, 486, 546);
                     // user code begin {1}
                     // user code end
              } catch ( Throwable ivjExc) {
                     // user code begin {2}
                     // user code end
                     handleException(ivjExc);
              }
       }
       return ivjConsultantFormPanel;
}
/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getConsultantFormPanel(), 
getConsultantFormPanel().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddConsultantDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(504, 566);
		setTitle("Add Consultant");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		AddConsultantDialog aAddConsultantDialog;
		aAddConsultantDialog = new AddConsultantDialog();
		aAddConsultantDialog.setModal(true);
		aAddConsultantDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aAddConsultantDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
	}
}
}

/*
 * Revision History
 *
 */
```

AddMachineDialog.java

```
//---------------------------------------------------------------------------
// AddMachineDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * A dialog class that is used with the addition of a machine.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the machine.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddMachineDialog extends  JDialog {
       private  JPanel ivjJDialogContentPane = null;
       private MachineFormPanel ivjMachineFormPanel = null;
/**
 * AddMachineDialog constructor comment.
 */
public AddMachineDialog() {
       super();
       initialize();
}
/**
 * AddMachineDialog constructor comment.
 * @param owner  Frame
 */
public AddMachineDialog( Frame owner) {
       super(owner);
}
/**
 * AddMachineDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddMachineDialog( Frame owner, String title) {
       super(owner, title);
}
/**
 * AddMachineDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddMachineDialog( Frame owner, String title, boolean modal) {
       super(owner, title, modal);
}
```

```
/**
 * AddMachineDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddMachineDialog( Frame owner, boolean modal) {
	super(owner, modal);
}
/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
				getJDialogContentPane().add(getMachineFormPanel(),
	getMachineFormPanel().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}
/**
 * Return the MachineFormPanel property value.
 * @return MachineFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private MachineFormPanel getMachineFormPanel() {
	if (ivjMachineFormPanel == null) {
		try {
			ivjMachineFormPanel = new MachineFormPanel();
			ivjMachineFormPanel.setName("MachineFormPanel");
			ivjMachineFormPanel.setBounds(5, 5, 545, 654);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjMachineFormPanel;
}
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddMachineDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(562, 658);
		setTitle("Add Machine");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		AddMachineDialog aAddMachineDialog;
		aAddMachineDialog = new AddMachineDialog();
		aAddMachineDialog.setModal(true);
		aAddMachineDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aAddMachineDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
	}
}
}

/*
 * Revision History
 *
 */
```

AddOfficeDialog.java

```
//------------------------------------------------------------------------
// AddOfficeDialog.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//------------------------------------------------------------------------

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


/**
 * A dialog class that is used with the addition of an office.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the office.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddOfficeDialog extends  JDialog {
// The private data members
        private  JLabel ivjCityLabel = null;
        private  JTextField ivjCityTextField = null;
        private  JLabel ivjCountryLabel = null;
        private  JTextField ivjCountryTextField = null;
        private  JPanel ivjInfoPanel = null;
        private  JPanel ivjJDialogContentPane = null;
        private ModifyButtonsPanel ivjModifyButtonsPanel = null;
        private  JLabel ivjNameLabel = null;
        private  JTextField ivjNameTextField = null;
        private  JLabel ivjStateLabel = null;
        private  JTextField ivjStateTextField = null;

/**
 * Calls the base class constructor and the initialize function.
 *
 **/
public AddOfficeDialog() {
        super();
        initialize();
}

/**
 * AddOfficeDialog constructor comment.
 *
 * @param owner  Frame
 **/
public AddOfficeDialog(Frame owner) {
        super(owner);
}

/**
 * AddOfficeDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 **/
public AddOfficeDialog( Frame owner, String title) {
        super(owner, title);
}

/**
 * AddOfficeDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 **/
public AddOfficeDialog( Frame owner, String title, boolean modal) {
        super(owner, title, modal);
}

/**
 * AddOfficeDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 **/
public AddOfficeDialog( Frame owner, boolean modal) {
        super(owner, modal);
}

/**
 * Return the CityLabel property value.
 *
 * @return  JLabel
 **/
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getCityLabel() {
        if (ivjCityLabel == null) {
                try {
```

```
				ivjCityLabel = new  JLabel();
				ivjCityLabel.setName("CityLabel");
				ivjCityLabel.setText("City");
				ivjCityLabel.setBounds(23, 37, 45, 15);
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjCityLabel;
}

/**
 * Return the CityTextField property value.
 *
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getCityTextField() {
		if (ivjCityTextField == null) {
			try {
				ivjCityTextField = new  JTextField();
				ivjCityTextField.setName("CityTextField");
				ivjCityTextField.setBounds(99, 35, 196, 19);
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjCityTextField;
}
/**
 * Return the CountryLabel property value.
 *
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getCountryLabel() {
		if (ivjCountryLabel == null) {
			try {
				ivjCountryLabel = new  JLabel();
				ivjCountryLabel.setName("CountryLabel");
				ivjCountryLabel.setText("Country");
				ivjCountryLabel.setBounds(23, 89, 45, 15);
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjCountryLabel;
}

/**
 * Return the CountryTextField property value.
 *
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getCountryTextField() {
		if (ivjCountryTextField == null) {
			try {
				ivjCountryTextField = new  JTextField();
```

```
                        ivjCountryTextField.setName("CountryTextField");
                        ivjCountryTextField.setBounds(99, 89, 196, 19);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjCountryTextField;
}

/**
 * Return the InfoPanel property value.
 *
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getInfoPanel() {
        if (ivjInfoPanel == null) {
                try {
                        ivjInfoPanel = new  JPanel();
                        ivjInfoPanel.setName("InfoPanel");
                        ivjInfoPanel.setLayout(null);
                        ivjInfoPanel.setBounds(54, 22, 318, 116);
                        getInfoPanel().add(getCityLabel(), getCityLabel().getName());
                        getInfoPanel().add(getCityTextField(), getCityTextField().getName());
                        getInfoPanel().add(getStateLabel(), getStateLabel().getName());
                        getInfoPanel().add(getStateTextField(), getStateTextField().getName());
                        getInfoPanel().add(getCountryLabel(), getCountryLabel().getName());
                        getInfoPanel().add(getCountryTextField(),
                        getCountryTextField().getName());
                        getInfoPanel().add(getNameLabel(), getNameLabel().getName());
                        getInfoPanel().add(getNameTextField(), getNameTextField().getName());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjInfoPanel;
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
        if (ivjJDialogContentPane == null) {
                try {
                        ivjJDialogContentPane = new  JPanel();
                        ivjJDialogContentPane.setName("JDialogContentPane");
                        ivjJDialogContentPane.setLayout(null);
                        getJDialogContentPane().add(getInfoPanel(), getInfoPanel().getName());
                        getJDialogContentPane().add(getModifyButtonsPanel(),
                        getModifyButtonsPanel().getName());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjJDialogContentPane;
}
```

```
/**
 * Return the ModifyButtonsPanel property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
	if (ivjModifyButtonsPanel == null) {
		try {
			ivjModifyButtonsPanel = new ModifyButtonsPanel();
			ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
			ivjModifyButtonsPanel.setBounds(92, 160, 242, 56);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyButtonsPanel;
}

/**
 * Return the NameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
	if (ivjNameLabel == null) {
		try {
			ivjNameLabel = new  JLabel();
			ivjNameLabel.setName("NameLabel");
			ivjNameLabel.setText("Name");
			ivjNameLabel.setBounds(22, 11, 45, 15);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameLabel;
}

/**
 * Return the NameTextField property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
	if (ivjNameTextField == null) {
		try {
			ivjNameTextField = new  JTextField();
			ivjNameTextField.setName("NameTextField");
			ivjNameTextField.setBounds(99, 8, 196, 19);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameTextField;
}

/**
 * Return the StateLabel property value.
 *
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getStateLabel() {
	if (ivjStateLabel == null) {
		try {
			ivjStateLabel = new  JLabel();
			ivjStateLabel.setName("StateLabel");
			ivjStateLabel.setText("State");
			ivjStateLabel.setBounds(23, 63, 45, 15);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjStateLabel;
}

/**
 * Return the StateTextField property value.
 *
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getStateTextField() {
	if (ivjStateTextField == null) {
		try {
			ivjStateTextField = new  JTextField();
			ivjStateTextField.setName("StateTextField");
			ivjStateTextField.setBounds(99, 62, 196, 19);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjStateTextField;
}

/**
 * Called whenever the part throws an exception.
 *
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {
	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddOfficeDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(426, 240);
		setTitle("Add Office");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
```

```
		// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * Only used to see what the dialog looks like until it is inserted into a main
 *
 * @param args  String[]
 */
public static void main( String[] args) {
		try {
			AddOfficeDialog aAddOfficeDialog;
			aAddOfficeDialog = new AddOfficeDialog();
			aAddOfficeDialog.setModal(true);
			aAddOfficeDialog.addWindowListener(new  WindowAdapter() {
				public void windowClosing( WindowEvent e) {
					System.exit(0);
				};
			});
			aAddOfficeDialog.setVisible(true);
		} catch (Throwable exception) {
			System.err.println("Exception occurred in main() of  JDialog");
			exception.printStackTrace(System.out);
		}
}
}

/*
 * Revision History
 *
 */
```

AddProjectDialog.java

```
//------------------------------------------------------------------------------
// AddProjectDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//------------------------------------------------------------------------------

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


/**
 * A dialog class that is used with the addition of a project.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the project.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddProjectDialog extends  JDialog {
		private  JPanel ivjJDialogContentPane = null;
		private ProjectFormPanel ivjProjectFormPanel = null;
/**
```

```
 * AddProjectDialog constructor comment.
 */
public AddProjectDialog() {
	super();
	initialize();
}
/**
 * AddProjectDialog constructor comment.
 * @param owner  Frame
 */
public AddProjectDialog( Frame owner) {
	super(owner);
}
/**
 * AddProjectDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddProjectDialog( Frame owner, String title) {
	super(owner, title);
}
/**
 * AddProjectDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddProjectDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}
/**
 * AddProjectDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddProjectDialog( Frame owner, boolean modal) {
	super(owner, modal);
}
/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getProjectFormPanel(),
			getProjectFormPanel().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}
/**
 * Return the ProjectFormPanel property value.
 * @return ProjectFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ProjectFormPanel getProjectFormPanel() {
	if (ivjProjectFormPanel == null) {
		try {
			ivjProjectFormPanel = new ProjectFormPanel();
			ivjProjectFormPanel.setName("ProjectFormPanel");
```

```
                        ivjProjectFormPanel.setLocation(6, 5);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjProjectFormPanel;
}
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("AddProjectDialog");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setSize(454, 354);
                setTitle("Add Project");
                setContentPane(getJDialogContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                AddProjectDialog aAddProjectDialog;
                aAddProjectDialog = new AddProjectDialog();
                aAddProjectDialog.setModal(true);
                aAddProjectDialog.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aAddProjectDialog.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JDialog");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

AddResourceAllocDialog.java

```
//----------------------------------------------------------------------------
// AddResourceAllocDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//----------------------------------------------------------------------------

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


/**
 * A dialog class that is used with the addition of a resource allocation.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the resource allocation.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddResourceAllocDialog extends  JDialog {
// Private data members
       private  JLabel ivjClientLabel = null;
       private  JPanel ivjJDialogContentPane = null;
       private  JLabel ivjProjectLabel = null;
       private  JComboBox ivjResourceNameComboBox = null;
       private  JLabel ivjResourceNameLabel = null;
       private  JComboBox ivjResourceTypeComboBox = null;
       private  JLabel ivjResourceTypeLabel = null;
       private  JPanel ivjSelectionPanel = null;
       private  JTextField ivjextraTextField = null;
       private  JTextField ivjJTextField1 = null;
       private  JTextField ivjJTextField2 = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel1 = null;

/**
 * AddResourceAllocDialog constructor comment.
 */
public AddResourceAllocDialog() {
       super();
       initialize();
}

/**
 * AddResourceAllocDialog constructor comment.
 * @param owner  Frame
 */
public AddResourceAllocDialog( Frame owner) {
       super(owner);
}

/**
 * AddResourceAllocDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddResourceAllocDialog( Frame owner, String title) {
       super(owner, title);
}
```

```
/**
 * AddResourceAllocDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddResourceAllocDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}

/**
 * AddResourceAllocDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddResourceAllocDialog( Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the ClientLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getClientLabel() {
	if (ivjClientLabel == null) {
		try {
			ivjClientLabel = new  JLabel();
			ivjClientLabel.setName("ClientLabel");
			ivjClientLabel.setText("Client");
			ivjClientLabel.setEnabled(true);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjClientLabel;
}

/**
 * Return the extraTextField property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getextraTextField() {
	if (ivjextraTextField == null) {
		try {
			ivjextraTextField = new  JTextField();
			ivjextraTextField.setName("extraTextField");
			ivjextraTextField.setEditable(false);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjextraTextField;
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getSelectionPanel(),
getSelectionPanel().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getJTextField1() {
	if (ivjJTextField1 == null) {
		try {
			ivjJTextField1 = new  JTextField();
			ivjJTextField1.setName("JTextField1");
			ivjJTextField1.setEnabled(true);
			ivjJTextField1.setEditable(false);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJTextField1;
}

/**
 * Return the JTextField2 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getJTextField2() {
	if (ivjJTextField2 == null) {
		try {
			ivjJTextField2 = new  JTextField();
			ivjJTextField2.setName("JTextField2");
			ivjJTextField2.setEditable(false);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJTextField2;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel1() {
	if (ivjModifyButtonsPanel1 == null) {
		try {
				ivjModifyButtonsPanel1 = new ModifyButtonsPanel();
				ivjModifyButtonsPanel1.setName("ModifyButtonsPanel1");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjModifyButtonsPanel1;
}

/**
 * Return the ProjectLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getProjectLabel() {
		if (ivjProjectLabel == null) {
			try {
				ivjProjectLabel = new  JLabel();
				ivjProjectLabel.setName("ProjectLabel");
				ivjProjectLabel.setText("Project");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjProjectLabel;
}

/**
 * Return the ResourceNameComboBox property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getResourceNameComboBox() {
		if (ivjResourceNameComboBox == null) {
			try {
				ivjResourceNameComboBox = new  JComboBox();
				ivjResourceNameComboBox.setName("ResourceNameComboBox");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjResourceNameComboBox;
}

/**
 * Return the ResourceNameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getResourceNameLabel() {
		if (ivjResourceNameLabel == null) {
			try {
				ivjResourceNameLabel = new  JLabel();
				ivjResourceNameLabel.setName("ResourceNameLabel");
				ivjResourceNameLabel.setText("Resource Name");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
```

```
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjResourceNameLabel;
}

/**
 * Return the ResourceTypeComboBox property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getResourceTypeComboBox() {
        if (ivjResourceTypeComboBox == null) {
                try {
                        ivjResourceTypeComboBox = new  JComboBox();
                        ivjResourceTypeComboBox.setName("ResourceTypeComboBox");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjResourceTypeComboBox;
}

/**
 * Return the ResourceTypeLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getResourceTypeLabel() {
        if (ivjResourceTypeLabel == null) {
                try {
                        ivjResourceTypeLabel = new  JLabel();
                        ivjResourceTypeLabel.setName("ResourceTypeLabel");
                        ivjResourceTypeLabel.setText("Resource Type");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjResourceTypeLabel;
}

/**
 * Return the SelectionPanel property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getSelectionPanel() {
        if (ivjSelectionPanel == null) {
                try {
                        ivjSelectionPanel = new  JPanel();
                        ivjSelectionPanel.setName("SelectionPanel");
                        ivjSelectionPanel.setLayout(new  GridBagLayout());
                        ivjSelectionPanel.setBounds(6, 4, 319, 287);

                        GridBagConstraints constraintsClientLabel = new  GridBagConstraints();
                        constraintsClientLabel.gridx = 1; constraintsClientLabel.gridy = 1;
                        constraintsClientLabel.ipadx = 13;
                        constraintsClientLabel.insets = new  Insets(25, 28, 12, 1);
                        getSelectionPanel().add(getClientLabel(), constraintsClientLabel);

                        GridBagConstraints constraintsProjectLabel = new  GridBagConstraints();
                        constraintsProjectLabel.gridx = 1; constraintsProjectLabel.gridy = 2;
                        constraintsProjectLabel.ipadx = 4;
                        constraintsProjectLabel.insets = new  Insets(13, 28, 12, 1);
                        getSelectionPanel().add(getProjectLabel(), constraintsProjectLabel);

                        GridBagConstraints constraintsResourceTypeLabel = new
                        GridBagConstraints();
                        constraintsResourceTypeLabel.gridx = 1; constraintsResourceTypeLabel.gridy
                        = 3;
                        constraintsResourceTypeLabel.gridwidth = 2;
                        constraintsResourceTypeLabel.ipadx = 12;
                        constraintsResourceTypeLabel.insets = new  Insets(13, 28, 17, 8);
                        getSelectionPanel().add(getResourceTypeLabel(),
                        constraintsResourceTypeLabel);

                        GridBagConstraints constraintsResourceNameLabel = new
                        GridBagConstraints();
                        constraintsResourceNameLabel.gridx = 1; constraintsResourceNameLabel.gridy
                        = 4;
                        constraintsResourceNameLabel.gridwidth = 2;
                        constraintsResourceNameLabel.ipadx = 10;
                        constraintsResourceNameLabel.insets = new  Insets(8, 28, 21, 4);
                        getSelectionPanel().add(getResourceNameLabel(),
                        constraintsResourceNameLabel);

                        GridBagConstraints constraintsResourceTypeComboBox = new
                        GridBagConstraints();
                        constraintsResourceTypeComboBox.gridx = 3;
                        constraintsResourceTypeComboBox.gridy = 3;
                        constraintsResourceTypeComboBox.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsResourceTypeComboBox.weightx = 1.0;
                        constraintsResourceTypeComboBox.ipadx = 12;
                        constraintsResourceTypeComboBox.insets = new  Insets(10, 5, 8, 43);
                        getSelectionPanel().add(getResourceTypeComboBox(),
                        constraintsResourceTypeComboBox);

                        GridBagConstraints constraintsResourceNameComboBox = new
                        GridBagConstraints();
                        constraintsResourceNameComboBox.gridx = 3;
                        constraintsResourceNameComboBox.gridy = 4;
                        constraintsResourceNameComboBox.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsResourceNameComboBox.weightx = 1.0;
                        constraintsResourceNameComboBox.ipadx = 12;
                        constraintsResourceNameComboBox.insets = new  Insets(8, 5, 9, 43);
                        getSelectionPanel().add(getResourceNameComboBox(),
                        constraintsResourceNameComboBox);

                        GridBagConstraints constraintsJTextField1 = new   GridBagConstraints();
                        constraintsJTextField1.gridx = 3; constraintsJTextField1.gridy = 1;
                        constraintsJTextField1.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsJTextField1.weightx = 1.0;
                        constraintsJTextField1.ipadx = 134;
                        constraintsJTextField1.insets = new  Insets(23, 5, 10, 43);
                        getSelectionPanel().add(getJTextField1(), constraintsJTextField1);

                        GridBagConstraints constraintsJTextField2 = new  GridBagConstraints();
                        constraintsJTextField2.gridx = 3; constraintsJTextField2.gridy = 2;
                        constraintsJTextField2.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsJTextField2.weightx = 1.0;
                        constraintsJTextField2.ipadx = 134;
                        constraintsJTextField2.insets = new  Insets(11, 5, 10, 43);
                        getSelectionPanel().add(getJTextField2(), constraintsJTextField2);

                        GridBagConstraints constraintsextraTextField = new  GridBagConstraints();
                        constraintsextraTextField.gridx = 2; constraintsextraTextField.gridy = 5;
                        constraintsextraTextField.gridwidth = 2;
                        constraintsextraTextField.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsextraTextField.weightx = 1.0;
                        constraintsextraTextField.ipadx = 196;
                        constraintsextraTextField.insets = new  Insets(10, 2, 5, 43);
                        getSelectionPanel().add(getextraTextField(), constraintsextraTextField);
```

```
                        GridBagConstraints constraintsModifyButtonsPanel1 = new
                        GridBagConstraints();
                        constraintsModifyButtonsPanel1.gridx = 1;
                        constraintsModifyButtonsPanel1.gridy = 6;
                        constraintsModifyButtonsPanel1.gridwidth = 3;
                        constraintsModifyButtonsPanel1.fill =  GridBagConstraints.BOTH;
                        constraintsModifyButtonsPanel1.weightx = 1.0;
                        constraintsModifyButtonsPanel1.weighty = 1.0;
                        constraintsModifyButtonsPanel1.insets = new  Insets(6, 35, 10, 42);
                        getSelectionPanel().add(getModifyButtonsPanel1(),
                        constraintsModifyButtonsPanel1);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjSelectionPanel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("AddResourceAllocDialog");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setSize(336, 298);
                setTitle("Add Resource Allocation");
                setContentPane(getJDialogContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                AddResourceAllocDialog aAddResourceAllocDialog;
                aAddResourceAllocDialog = new AddResourceAllocDialog();
                aAddResourceAllocDialog.setModal(true);
                aAddResourceAllocDialog.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aAddResourceAllocDialog.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JDialog");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

AddRoomDialog.java

```
//--------------------------------------------------------------------------------
// AddRoomDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//--------------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the addition of a room.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the room.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddRoomDialog extends  JDialog {
// Private data members
        private  JPanel ivjJDialogContentPane = null;
        private RoomFormPanel ivjRoomFormPanel = null;

/**
 * AddRoomDialog constructor comment.
 */
public AddRoomDialog() {
        super();
        initialize();
}

/**
 * AddRoomDialog constructor comment.
 * @param owner  Frame
 */
public AddRoomDialog( Frame owner) {
        super(owner);
}

/**
 * AddRoomDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddRoomDialog( Frame owner, String title) {
	super(owner, title);
}

/**
 * AddRoomDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddRoomDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}

/**
 * AddRoomDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddRoomDialog( Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(new  GridBagLayout());

			GridBagConstraints constraintsRoomFormPanel = new  GridBagConstraints();
			constraintsRoomFormPanel.gridx = 1; constraintsRoomFormPanel.gridy = 1;
			constraintsRoomFormPanel.fill =  GridBagConstraints.BOTH;
			constraintsRoomFormPanel.weightx = 1.0;
			constraintsRoomFormPanel.weighty = 1.0;
			constraintsRoomFormPanel.ipady = 33;
			constraintsRoomFormPanel.insets = new  Insets(9, 10, 4, 3);
			getJDialogContentPane().add(getRoomFormPanel(), constraintsRoomFormPanel);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the RoomFormPanel property value.
 * @return RoomFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private RoomFormPanel getRoomFormPanel() {
	if (ivjRoomFormPanel == null) {
		try {
			ivjRoomFormPanel = new RoomFormPanel();
			ivjRoomFormPanel.setName("RoomFormPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
```

```
        return ivjRoomFormPanel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("AddRoomDialog");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setSize(614, 610);
                setTitle("Add Room");
                setContentPane(getJDialogContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                AddRoomDialog aAddRoomDialog;
                aAddRoomDialog = new AddRoomDialog();
                aAddRoomDialog.setModal(true);
                aAddRoomDialog.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aAddRoomDialog.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JDialog");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

AddTimeslotDialog.java

```
//------------------------------------------------------------------------------
// AddTimeslotDialog.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
```

```
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//----------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the addition of a timeslot.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the timeslot.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class AddTimeslotDialog extends JDialog {
// Private data members
       private JPanel ivjJDialogContentPane = null;
       private TimeslotFormPanel ivjTimeslotFormPanel1 = null;

/**
 * AddTimeslotDialog constructor comment.
 */
public AddTimeslotDialog() {
       super();
       initialize();
}

/**
 * AddTimeslotDialog constructor comment.
 * @param owner Frame
 */
public AddTimeslotDialog(Frame owner) {
       super(owner);
}

/**
 * AddTimeslotDialog constructor comment.
 * @param owner Frame
 * @param title String
 */
public AddTimeslotDialog(Frame owner, String title) {
       super(owner, title);
}

/**
 * AddTimeslotDialog constructor comment.
 * @param owner Frame
 * @param title String
 * @param modal boolean
 */
public AddTimeslotDialog(Frame owner, String title, boolean modal) {
       super(owner, title, modal);
}

/**
 * AddTimeslotDialog constructor comment.
 * @param owner Frame
 * @param modal boolean
 */
public AddTimeslotDialog(Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the JDialogContentPane property value.
 * @return JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getTimeslotFormPanel1(), 
getTimeslotFormPanel1().getName());
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the TimeslotFormPanel1 property value.
 * @return TimeslotFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TimeslotFormPanel getTimeslotFormPanel1() {
	if (ivjTimeslotFormPanel1 == null) {
		try {
			ivjTimeslotFormPanel1 = new TimeslotFormPanel();
			ivjTimeslotFormPanel1.setName("TimeslotFormPanel1");
			ivjTimeslotFormPanel1.setLocation(6, 4);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjTimeslotFormPanel1;
}

/**
 * Called whenever the part throws an exception.
 * @param exception Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddTimeslotDialog");
		setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
		setSize(586, 448);
```

```
			setTitle("Add Timeslot");
			setContentPane(getJDialogContentPane());
		} catch (Throwable ivjExc) {
			handleException(ivjExc);
		}
		// user code begin {2}
		// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args String[]
 */
public static void main(String[] args) {
		try {
			AddTimeslotDialog aAddTimeslotDialog;
			aAddTimeslotDialog = new AddTimeslotDialog();
			aAddTimeslotDialog.setModal(true);
			aAddTimeslotDialog.addWindowListener(new WindowAdapter() {
				public void windowClosing(WindowEvent e) {
					System.exit(0);
				};
			});
			aAddTimeslotDialog.setVisible(true);
		} catch (Throwable exception) {
			System.err.println("Exception occurred in main() of JDialog");
			exception.printStackTrace(System.out);
		}
}
}

/*
 * Revision History
 *
 */
```

AddUserDialog.java

```
//---------------------------------------------------------------------------
// AddUserDialog.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the addition of a user.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the addition of the user.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/
class AddUserDialog extends  JDialog {
        private  JPanel ivjJDialogContentPane = null;
        private UserFormPanel ivjUserFormPanel = null;

/**
 * AddUserDialog constructor comment.
 */
public AddUserDialog() {
        super();
        initialize();
}

/**
 * AddUserDialog constructor comment.
 * @param owner  Frame
 */
public AddUserDialog( Frame owner) {
        super(owner);
}

/**
 * AddUserDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public AddUserDialog( Frame owner, String title) {
        super(owner, title);
}

/**
 * AddUserDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public AddUserDialog( Frame owner, String title, boolean modal) {
        super(owner, title, modal);
}

/**
 * AddUserDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public AddUserDialog( Frame owner, boolean modal) {
        super(owner, modal);
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
        if (ivjJDialogContentPane == null) {
                try {
                        ivjJDialogContentPane = new  JPanel();
                        ivjJDialogContentPane.setName("JDialogContentPane");
                        ivjJDialogContentPane.setLayout(null);
                        getJDialogContentPane().add(getUserFormPanel(),
                        getUserFormPanel().getName());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjJDialogContentPane;
}
```

```
/**
 * Return the UserFormPanel property value.
 * @return UserFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private UserFormPanel getUserFormPanel() {
	if (ivjUserFormPanel == null) {
		try {
			ivjUserFormPanel = new UserFormPanel();
			ivjUserFormPanel.setName("UserFormPanel");
			ivjUserFormPanel.setLocation(6, 6);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjUserFormPanel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AddUserDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(470, 348);
		setTitle("Add User");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		AddUserDialog aAddUserDialog;
		aAddUserDialog = new AddUserDialog();
		aAddUserDialog.setModal(true);
		aAddUserDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aAddUserDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
```

```
        }
}
}

/*
 * Revision History
 *
 */
```

ChangePasswordDialog.java

```
//-----------------------------------------------------------------------------
// ChangePasswordDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/29/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//-----------------------------------------------------------------------------

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

/**
 * A dialog class that is used with the changing of the users password.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the old password and what the new password should be.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class ChangePasswordDialog extends  JDialog {
        private  JButton ivjCancelButtton = null;
        private  JButton ivjChangeButton = null;
        private  JPasswordField ivjConfirmPasswordField = null;
        private  JLabel ivjConfirmPasswordLabel = null;
        private  JPanel ivjJDialogContentPane = null;
        private  JPasswordField ivjNewPasswordField = null;
        private  JLabel ivjNewPasswordLabel = null;
        private  JPasswordField ivjOldPasswordField = null;
        private  JLabel ivjOldPasswordLabel = null;
/**
 * ChangePasswordDialog constructor comment.
 */
public ChangePasswordDialog() {
        super();
        initialize();
}
/**
 * ChangePasswordDialog constructor comment.
 * @param owner  Frame
 */
public ChangePasswordDialog( Frame owner) {
        super(owner);
}
/**
 * ChangePasswordDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public ChangePasswordDialog( Frame owner, String title) {
	super(owner, title);
}
/**
 * ChangePasswordDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public ChangePasswordDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}
/**
 * ChangePasswordDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public ChangePasswordDialog( Frame owner, boolean modal) {
	super(owner, modal);
}
/**
 * Return the CancelButtton property value.
 * @return  JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JButton getCancelButtton() {
	if (ivjCancelButtton == null) {
		try {
			ivjCancelButtton = new  JButton();
			ivjCancelButtton.setName("CancelButtton");
			ivjCancelButtton.setText("Cancel");
			ivjCancelButtton.setBounds(261, 181, 73, 25);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjCancelButtton;
}
/**
 * Return the ChangeButton property value.
 * @return  JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JButton getChangeButton() {
	if (ivjChangeButton == null) {
		try {
			ivjChangeButton = new  JButton();
			ivjChangeButton.setName("ChangeButton");
			ivjChangeButton.setText("Change");
			ivjChangeButton.setBounds(92, 181, 77, 25);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjChangeButton;
}
/**
 * Return the ConfirmPasswordField property value.
 * @return  JPasswordField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
private  JPasswordField getConfirmPasswordField() {
	if (ivjConfirmPasswordField == null) {
		try {
			ivjConfirmPasswordField = new  JPasswordField();
			ivjConfirmPasswordField.setName("ConfirmPasswordField");
			ivjConfirmPasswordField.setBounds(182, 131, 145, 19);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjConfirmPasswordField;
}
/**
 * Return the ConfirmPasswordLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getConfirmPasswordLabel() {
	if (ivjConfirmPasswordLabel == null) {
		try {
			ivjConfirmPasswordLabel = new  JLabel();
			ivjConfirmPasswordLabel.setName("ConfirmPasswordLabel");
			ivjConfirmPasswordLabel.setText("Confirm Password");
			ivjConfirmPasswordLabel.setBounds(34, 132, 106, 15);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjConfirmPasswordLabel;
}
/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getOldPasswordLabel(),
			getOldPasswordLabel().getName());
			getJDialogContentPane().add(getOldPasswordField(),
			getOldPasswordField().getName());
			getJDialogContentPane().add(getNewPasswordLabel(),
			getNewPasswordLabel().getName());
			getJDialogContentPane().add(getNewPasswordField(),
			getNewPasswordField().getName());
			getJDialogContentPane().add(getConfirmPasswordLabel(),
			getConfirmPasswordLabel().getName());
			getJDialogContentPane().add(getConfirmPasswordField(),
			getConfirmPasswordField().getName());
			getJDialogContentPane().add(getChangeButton(),
			getChangeButton().getName());
			getJDialogContentPane().add(getCancelButtton(),
			getCancelButtton().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
```

```
                }
        }
        return ivjJDialogContentPane;
}
/**
 * Return the NewPasswordField property value.
 * @return  JPasswordField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPasswordField getNewPasswordField() {
        if (ivjNewPasswordField == null) {
                try {
                        ivjNewPasswordField = new  JPasswordField();
                        ivjNewPasswordField.setName("NewPasswordField");
                        ivjNewPasswordField.setBounds(182, 81, 145, 19);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNewPasswordField;
}
/**
 * Return the NewPasswordLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNewPasswordLabel() {
        if (ivjNewPasswordLabel == null) {
                try {
                        ivjNewPasswordLabel = new  JLabel();
                        ivjNewPasswordLabel.setName("NewPasswordLabel");
                        ivjNewPasswordLabel.setText("New Password");
                        ivjNewPasswordLabel.setBounds(34, 83, 86, 15);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNewPasswordLabel;
}
/**
 * Return the OldPasswordField property value.
 * @return  JPasswordField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPasswordField getOldPasswordField() {
        if (ivjOldPasswordField == null) {
                try {
                        ivjOldPasswordField = new  JPasswordField();
                        ivjOldPasswordField.setName("OldPasswordField");
                        ivjOldPasswordField.setBounds(182, 31, 145, 19);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjOldPasswordField;
}
/**
 * Return the OldPasswordLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getOldPasswordLabel() {
	if (ivjOldPasswordLabel == null) {
		try {
			ivjOldPasswordLabel = new  JLabel();
			ivjOldPasswordLabel.setName("OldPasswordLabel");
			ivjOldPasswordLabel.setText("Old Password");
			ivjOldPasswordLabel.setBounds(34, 34, 80, 15);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjOldPasswordLabel;
}
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("ChangePasswordDialog");
		setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
		setSize(426, 240);
		setTitle("Change Password");
		setContentPane(getJDialogContentPane());
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		ChangePasswordDialog aChangePasswordDialog;
		aChangePasswordDialog = new ChangePasswordDialog();
		aChangePasswordDialog.setModal(true);
		aChangePasswordDialog.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		aChangePasswordDialog.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JDialog");
		exception.printStackTrace(System.out);
	}
}
}

/*
```

```
 * Revision History
 *
 */
```

ClientFormPanel.java

```
//----------------------------------------------------------------------------
// ClientFormPanel.java
//
// Author: Mark Briggs
//                            John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//----------------------------------------------------------------------------

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

/**
 * A panel class that is used with the client information.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modificaion of the client information.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class ClientFormPanel extends  JPanel {
       private  JTextField ivjBusinessDevTextField = null;
       private  JTextField ivjCompanyNameTextField = null;
       private  JCheckBox ivjCSACheckBox = null;
       private  JLabel ivjBusinessDevLabel = null;
       private  JLabel ivjCompanyNameLabel = null;
       private  JLabel ivjHeadingLabel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private  JLabel ivjPrincipalLabel = null;
       private  JTextField ivjPrincipalTextField = null;
/**
 * ClientFormPanel constructor comment.
 */
public ClientFormPanel() {
       super();
       initialize();
}
/**
 * ClientFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public ClientFormPanel( LayoutManager layout) {
       super(layout);
}
/**
 * ClientFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public ClientFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
       super(layout, isDoubleBuffered);
}
```

```
/**
 * ClientFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public ClientFormPanel(boolean isDoubleBuffered) {
	super(isDoubleBuffered);
}
/**
 * Return the JLabel4 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getBusinessDevLabel() {
	if (ivjBusinessDevLabel == null) {
		try {
			ivjBusinessDevLabel = new  JLabel();
			ivjBusinessDevLabel.setName("BusinessDevLabel");
			ivjBusinessDevLabel.setText("Business Development Contact");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjBusinessDevLabel;
}
/**
 * Return the JTextField3 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getBusinessDevTextField() {
	if (ivjBusinessDevTextField == null) {
		try {
			ivjBusinessDevTextField = new  JTextField();
			ivjBusinessDevTextField.setName("BusinessDevTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjBusinessDevTextField;
}
/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getCompanyNameLabel() {
	if (ivjCompanyNameLabel == null) {
		try {
			ivjCompanyNameLabel = new  JLabel();
			ivjCompanyNameLabel.setName("CompanyNameLabel");
			ivjCompanyNameLabel.setText("Company Name");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjCompanyNameLabel;
}
/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getCompanyNameTextField() {
	if (ivjCompanyNameTextField == null) {
		try {
			ivjCompanyNameTextField = new  JTextField();
			ivjCompanyNameTextField.setName("CompanyNameTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjCompanyNameTextField;
}
/**
 * Return the JCheckBox1 property value.
 * @return  JCheckBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JCheckBox getCSACheckBox() {
	if (ivjCSACheckBox == null) {
		try {
			ivjCSACheckBox = new  JCheckBox();
			ivjCSACheckBox.setName("CSACheckBox");
			ivjCSACheckBox.setText("Is the CSA signed?");
			ivjCSACheckBox.setForeground(new  Color(102,102,153));
			ivjCSACheckBox.setHorizontalTextPosition( SwingConstants.LEFT);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjCSACheckBox;
}
/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
	if (ivjHeadingLabel == null) {
		try {
			ivjHeadingLabel = new  JLabel();
			ivjHeadingLabel.setName("HeadingLabel");
			ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
			ivjHeadingLabel.setText("Client Information");
			ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjHeadingLabel;
}
/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
	if (ivjModifyButtonsPanel == null) {
		try {
			ivjModifyButtonsPanel = new ModifyButtonsPanel();
			ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyButtonsPanel;
}
/**
 * Return the JLabel3 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getPrincipalLabel() {
	if (ivjPrincipalLabel == null) {
		try {
			ivjPrincipalLabel = new  JLabel();
			ivjPrincipalLabel.setName("PrincipalLabel");
			ivjPrincipalLabel.setText("Principal Contact Person");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjPrincipalLabel;
}
/**
 * Return the JTextField2 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getPrincipalTextField() {
	if (ivjPrincipalTextField == null) {
		try {
			ivjPrincipalTextField = new  JTextField();
			ivjPrincipalTextField.setName("PrincipalTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjPrincipalTextField;
}
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
// user code end
setName("ClientFormPanel");
setLayout(new  GridBagLayout());
setSize(526, 368);

GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
constraintsHeadingLabel.gridx = 1; constraintsHeadingLabel.gridy = 1;
constraintsHeadingLabel.gridwidth = 2;
constraintsHeadingLabel.ipadx = 122;
constraintsHeadingLabel.ipady = 24;
constraintsHeadingLabel.insets = new  Insets(22, 98, 26, 99);
add(getHeadingLabel(), constraintsHeadingLabel);

GridBagConstraints constraintsCompanyNameLabel = new  GridBagConstraints();
constraintsCompanyNameLabel.gridx = 1; constraintsCompanyNameLabel.gridy = 2;
constraintsCompanyNameLabel.ipadx = 28;
constraintsCompanyNameLabel.insets = new  Insets(28, 66, 13, 57);
add(getCompanyNameLabel(), constraintsCompanyNameLabel);

GridBagConstraints constraintsCompanyNameTextField = new  GridBagConstraints();
constraintsCompanyNameTextField.gridx = 2; constraintsCompanyNameTextField.gridy =
2;
constraintsCompanyNameTextField.fill =  GridBagConstraints.HORIZONTAL;
constraintsCompanyNameTextField.weightx = 1.0;
constraintsCompanyNameTextField.ipadx = 218;
constraintsCompanyNameTextField.insets = new  Insets(26, 12, 11, 52);
add(getCompanyNameTextField(), constraintsCompanyNameTextField);

GridBagConstraints constraintsPrincipalLabel = new  GridBagConstraints();
constraintsPrincipalLabel.gridx = 1; constraintsPrincipalLabel.gridy = 3;
constraintsPrincipalLabel.ipadx = 14;
constraintsPrincipalLabel.insets = new  Insets(13, 66, 10, 19);
add(getPrincipalLabel(), constraintsPrincipalLabel);

GridBagConstraints constraintsPrincipalTextField = new  GridBagConstraints();
constraintsPrincipalTextField.gridx = 2; constraintsPrincipalTextField.gridy = 3;
constraintsPrincipalTextField.fill =  GridBagConstraints.HORIZONTAL;
constraintsPrincipalTextField.weightx = 1.0;
constraintsPrincipalTextField.ipadx = 218;
constraintsPrincipalTextField.insets = new  Insets(11, 14, 8, 50);
add(getPrincipalTextField(), constraintsPrincipalTextField);

GridBagConstraints constraintsBusinessDevLabel = new  GridBagConstraints();
constraintsBusinessDevLabel.gridx = 1; constraintsBusinessDevLabel.gridy = 4;
constraintsBusinessDevLabel.gridwidth = 2;
constraintsBusinessDevLabel.ipadx = 18;
constraintsBusinessDevLabel.insets = new  Insets(11, 66, 12, 265);
add(getBusinessDevLabel(), constraintsBusinessDevLabel);

GridBagConstraints constraintsBusinessDevTextField = new  GridBagConstraints();
constraintsBusinessDevTextField.gridx = 2; constraintsBusinessDevTextField.gridy =
4;
constraintsBusinessDevTextField.fill =  GridBagConstraints.HORIZONTAL;
constraintsBusinessDevTextField.weightx = 1.0;
constraintsBusinessDevTextField.ipadx = 218;
constraintsBusinessDevTextField.insets = new  Insets(9, 14, 10, 50);
add(getBusinessDevTextField(), constraintsBusinessDevTextField);

GridBagConstraints constraintsCSACheckBox = new  GridBagConstraints();
constraintsCSACheckBox.gridx = 1; constraintsCSACheckBox.gridy = 5;
constraintsCSACheckBox.ipadx = 30;
constraintsCSACheckBox.insets = new  Insets(11, 66, 6, 11);
add(getCSACheckBox(), constraintsCSACheckBox);

GridBagConstraints constraintsModifyButtonsPanel = new  GridBagConstraints();
constraintsModifyButtonsPanel.gridx = 1; constraintsModifyButtonsPanel.gridy = 6;
constraintsModifyButtonsPanel.gridwidth = 2;
constraintsModifyButtonsPanel.fill =  GridBagConstraints.BOTH;
constraintsModifyButtonsPanel.weightx = 1.0;
constraintsModifyButtonsPanel.weighty = 1.0;
constraintsModifyButtonsPanel.insets = new  Insets(6, 142, 31, 142);
```

```
                add(getModifyButtonsPanel(), constraintsModifyButtonsPanel);
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}


/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                 JFrame frame = new  JFrame();
                ClientFormPanel aClientFormPanel;
                aClientFormPanel = new ClientFormPanel();
                frame.setContentPane(aClientFormPanel);
                frame.setSize(aClientFormPanel.getSize());
                frame.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                frame.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JPanel");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

ConsultantFormPanel.java

```
//--------------------------------------------------------------------------------
// ConsultantFormPanel.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/19/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//--------------------------------------------------------------------------------

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

/**
 * A panel class that is used with consultant information.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modification of teh consultant information.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/
```

```
class ConsultantFormPanel extends  JPanel {
// Private data members
       private DefaultRatePanel ivjDefaultRatePanel = null;
       private  JLabel ivjDefaultsLabel = null;
       private  JLabel ivjEmailAddressLabel = null;
       private  JTextField ivjEmailAddressTextField = null;
       private  JLabel ivjHeadingLabel = null;
       private  JPanel ivjInfoPanel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private  JLabel ivjNameLabel = null;
       private  JTextField ivjNameTextField = null;

/**
 * ConsultantFormPanel constructor comment.
 */
public ConsultantFormPanel() {
       super();
       initialize();
}

/**
 * ConsultantFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public ConsultantFormPanel( LayoutManager layout) {
       super(layout);
}

/**
 * ConsultantFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public ConsultantFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
       super(layout, isDoubleBuffered);
}

/**
 * ConsultantFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public ConsultantFormPanel(boolean isDoubleBuffered) {
       super(isDoubleBuffered);
}

/**
 * Return the DefaultRatePanel1 property value.
 * @return DefaultRatePanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private DefaultRatePanel getDefaultRatePanel() {
       if (ivjDefaultRatePanel == null) {
              try {
                     ivjDefaultRatePanel = new DefaultRatePanel();
                     ivjDefaultRatePanel.setName("DefaultRatePanel");
                     // user code begin {1}
                     // user code end
              } catch ( Throwable ivjExc) {
                     // user code begin {2}
                     // user code end
                     handleException(ivjExc);
              }
       }
       return ivjDefaultRatePanel;
}

/**
 * Return the JLabel3 property value.
 * @return  JLabel
 */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getDefaultsLabel() {
	if (ivjDefaultsLabel == null) {
		try {
			ivjDefaultsLabel = new  JLabel();
			ivjDefaultsLabel.setName("DefaultsLabel");
			ivjDefaultsLabel.setText("Defaults");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjDefaultsLabel;
}

/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getEmailAddressLabel() {
	if (ivjEmailAddressLabel == null) {
		try {
			ivjEmailAddressLabel = new  JLabel();
			ivjEmailAddressLabel.setName("EmailAddressLabel");
			ivjEmailAddressLabel.setText("E-mail Address");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjEmailAddressLabel;
}

/**
 * Return the JTextField2 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getEmailAddressTextField() {
	if (ivjEmailAddressTextField == null) {
		try {
			ivjEmailAddressTextField = new  JTextField();
			ivjEmailAddressTextField.setName("EmailAddressTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjEmailAddressTextField;
}

/**
 * Return the JLabel10 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
	if (ivjHeadingLabel == null) {
		try {
			ivjHeadingLabel = new  JLabel();
			ivjHeadingLabel.setName("HeadingLabel");
```

```
                        ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
                        ivjHeadingLabel.setText("Consultant Information");
                        ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHeadingLabel;
}

/**
 * Return the JPanel2 property value.  This panel is the information panel
 * and is used to gather the name and email address of the consultant
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getInfoPanel() {
        if (ivjInfoPanel == null) {
                try {
                        ivjInfoPanel = new  JPanel();
                        ivjInfoPanel.setName("InfoPanel");
                        ivjInfoPanel.setLayout(new  GridBagLayout());

                        GridBagConstraints constraintsNameTextField = new  GridBagConstraints();
                        constraintsNameTextField.gridx = 2; constraintsNameTextField.gridy = 1;
                        constraintsNameTextField.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsNameTextField.weightx = 1.0;
                        constraintsNameTextField.ipadx = 180;
                        constraintsNameTextField.insets = new  Insets(27, 8, 13, 76);
                        getInfoPanel().add(getNameTextField(), constraintsNameTextField);

                        GridBagConstraints constraintsEmailAddressTextField = new
                        GridBagConstraints();
                        constraintsEmailAddressTextField.gridx = 2;
                        constraintsEmailAddressTextField.gridy = 2;
                        constraintsEmailAddressTextField.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsEmailAddressTextField.weightx = 1.0;
                        constraintsEmailAddressTextField.ipadx = 180;
                        constraintsEmailAddressTextField.insets = new  Insets(14, 8, 28, 76);
                        getInfoPanel().add(getEmailAddressTextField(),
                        constraintsEmailAddressTextField);

                        GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
                        constraintsNameLabel.gridx = 1; constraintsNameLabel.gridy = 1;
                        constraintsNameLabel.ipadx = 12;
                        constraintsNameLabel.insets = new  Insets(29, 4, 15, 59);
                        getInfoPanel().add(getNameLabel(), constraintsNameLabel);

                        GridBagConstraints constraintsEmailAddressLabel = new
                        GridBagConstraints();
                        constraintsEmailAddressLabel.gridx = 1; constraintsEmailAddressLabel.gridy
                        = 2;
                        constraintsEmailAddressLabel.ipadx = 11;
                        constraintsEmailAddressLabel.insets = new  Insets(16, 4, 30, 7);
                        getInfoPanel().add(getEmailAddressLabel(), constraintsEmailAddressLabel);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjInfoPanel;
}

/**
```

```
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
	if (ivjModifyButtonsPanel == null) {
		try {
			ivjModifyButtonsPanel = new ModifyButtonsPanel();
			ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyButtonsPanel;
}

/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
	if (ivjNameLabel == null) {
		try {
			ivjNameLabel = new  JLabel();
			ivjNameLabel.setName("NameLabel");
			ivjNameLabel.setText("Name");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameLabel;
}

/**
 * Return the JTextField21 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
	if (ivjNameTextField == null) {
		try {
			ivjNameTextField = new  JTextField();
			ivjNameTextField.setName("NameTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameTextField;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("ConsultantFormPanel");
		setLayout(new  GridBagLayout());
		setSize(486, 546);

		GridBagConstraints constraintsDefaultsLabel = new  GridBagConstraints();
		constraintsDefaultsLabel.gridx = 1; constraintsDefaultsLabel.gridy = 3;
		constraintsDefaultsLabel.ipadx = 28;
		constraintsDefaultsLabel.insets = new  Insets(3, 44, 1, 1);
		add(getDefaultsLabel(), constraintsDefaultsLabel);

		GridBagConstraints constraintsInfoPanel = new  GridBagConstraints();
		constraintsInfoPanel.gridx = 1; constraintsInfoPanel.gridy = 2;
		constraintsInfoPanel.gridwidth = 2;
		constraintsInfoPanel.fill =  GridBagConstraints.BOTH;
		constraintsInfoPanel.weightx = 1.0;
		constraintsInfoPanel.weighty = 1.0;
		constraintsInfoPanel.ipady = -2;
		constraintsInfoPanel.insets = new  Insets(12, 42, 2, 68);
		add(getInfoPanel(), constraintsInfoPanel);

		GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
		constraintsHeadingLabel.gridx = 1; constraintsHeadingLabel.gridy = 1;
		constraintsHeadingLabel.gridwidth = 2;
		constraintsHeadingLabel.ipadx = 60;
		constraintsHeadingLabel.ipady = 38;
		constraintsHeadingLabel.insets = new  Insets(23, 80, 11, 80);
		add(getHeadingLabel(), constraintsHeadingLabel);

		GridBagConstraints constraintsDefaultRatePanel = new  GridBagConstraints();
		constraintsDefaultRatePanel.gridx = 1; constraintsDefaultRatePanel.gridy = 4;
		constraintsDefaultRatePanel.gridwidth = 2;
		constraintsDefaultRatePanel.fill =  GridBagConstraints.BOTH;
		constraintsDefaultRatePanel.weightx = 1.0;
		constraintsDefaultRatePanel.weighty = 1.0;
		constraintsDefaultRatePanel.ipadx = -4;
		constraintsDefaultRatePanel.ipady = -2;
		constraintsDefaultRatePanel.insets = new  Insets(2, 42, 5, 56);
		add(getDefaultRatePanel(), constraintsDefaultRatePanel);

		GridBagConstraints constraintsModifyButtonsPanel = new  GridBagConstraints();
		constraintsModifyButtonsPanel.gridx = 2; constraintsModifyButtonsPanel.gridy = 5;
		constraintsModifyButtonsPanel.fill =  GridBagConstraints.BOTH;
		constraintsModifyButtonsPanel.weightx = 1.0;
		constraintsModifyButtonsPanel.weighty = 1.0;
		constraintsModifyButtonsPanel.insets = new  Insets(5, 2, 10, 122);
		add(getModifyButtonsPanel(), constraintsModifyButtonsPanel);
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		 JFrame frame = new  JFrame();
```

```
               ConsultantFormPanel aConsultantFormPanel;
               aConsultantFormPanel = new ConsultantFormPanel();
               frame.setContentPane(aConsultantFormPanel);
               frame.setSize(aConsultantFormPanel.getSize());
               frame.addWindowListener(new  WindowAdapter() {
                      public void windowClosing( WindowEvent e) {
                             System.exit(0);
                      };
               });
               frame.setVisible(true);
        } catch (Throwable exception) {
               System.err.println("Exception occurred in main() of  JPanel");
               exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

DefaultRatePanel.java

```
//---------------------------------------------------------------------------
// DefaultRatePanel.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------

import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * A panel class that is used to hold the default information for all resource
 * types. It creates the necessary form that will be able to obtain all the
 * necessary information for the addition of the office.
 *
 * @version   1.1
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class DefaultRatePanel extends  JPanel {
// Private data members
        private  JLabel ivjOfficeLabel = null;
        private  JLabel ivjOvertimeCostLabel = null;
        private  JTextField ivjOvertimeCostTextField = null;
        private  JLabel ivjOvertimeRateLabel = null;
        private  JTextField ivjOvertimeRateTextField = null;
        private  JLabel ivjRegularCostLabel = null;
        private  JTextField ivjRegularCostTextField = null;
        private  JTextField ivjRegularRateTextField = null;
        private  JComboBox ivjOfficeComboBox = null;
        private  JLabel ivjtargetRateLabel = null;
```

```
/**
 * DefaultRatePanel constructor comment.
 */
public DefaultRatePanel() {
	super();
	initialize();
}

/**
 * DefaultRatePanel constructor comment.
 * @param layout  LayoutManager
 */
public DefaultRatePanel( LayoutManager layout) {
	super(layout);
}

/**
 * DefaultRatePanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public DefaultRatePanel( LayoutManager layout, boolean isDoubleBuffered) {
	super(layout, isDoubleBuffered);
}

/**
 * DefaultRatePanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public DefaultRatePanel(boolean isDoubleBuffered) {
	super(isDoubleBuffered);
}

/**
 * Return the OfficeComboBox property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getOfficeComboBox() {
	if (ivjOfficeComboBox == null) {
		try {
			ivjOfficeComboBox = new  JComboBox();
			ivjOfficeComboBox.setName("OfficeComboBox");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjOfficeComboBox;
}

/**
 * Return the JLabel9 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getOfficeLabel() {
	if (ivjOfficeLabel == null) {
		try {
			ivjOfficeLabel = new  JLabel();
			ivjOfficeLabel.setName("OfficeLabel");
			ivjOfficeLabel.setText("Office");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
```

```
				}
		}
		return ivjOfficeLabel;
}

/**
 * Return the JLabel7 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getOvertimeCostLabel() {
		if (ivjOvertimeCostLabel == null) {
				try {
						ivjOvertimeCostLabel = new  JLabel();
						ivjOvertimeCostLabel.setName("OvertimeCostLabel");
						ivjOvertimeCostLabel.setText("Overtime Cost");
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjOvertimeCostLabel;
}

/**
 * Return the JTextField113 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getOvertimeCostTextField() {
		if (ivjOvertimeCostTextField == null) {
				try {
						ivjOvertimeCostTextField = new  JTextField();
						ivjOvertimeCostTextField.setName("OvertimeCostTextField");
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjOvertimeCostTextField;
}

/**
 * Return the JLabel5 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getOvertimeRateLabel() {
		if (ivjOvertimeRateLabel == null) {
				try {
						ivjOvertimeRateLabel = new  JLabel();
						ivjOvertimeRateLabel.setName("OvertimeRateLabel");
						ivjOvertimeRateLabel.setText("Overtime Target Rate");
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjOvertimeRateLabel;
}

/**
```

```
 * Return the JTextField111 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getOvertimeRateTextField() {
	if (ivjOvertimeRateTextField == null) {
		try {
			ivjOvertimeRateTextField = new  JTextField();
			ivjOvertimeRateTextField.setName("OvertimeRateTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjOvertimeRateTextField;
}

/**
 * Return the JLabel6 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getRegularCostLabel() {
	if (ivjRegularCostLabel == null) {
		try {
			ivjRegularCostLabel = new  JLabel();
			ivjRegularCostLabel.setName("RegularCostLabel");
			ivjRegularCostLabel.setText("Regular Cost");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjRegularCostLabel;
}

/**
 * Return the JTextField112 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getRegularCostTextField() {
	if (ivjRegularCostTextField == null) {
		try {
			ivjRegularCostTextField = new  JTextField();
			ivjRegularCostTextField.setName("RegularCostTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjRegularCostTextField;
}

/**
 * Return the JTextField11 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getRegularRateTextField() {
	if (ivjRegularRateTextField == null) {
		try {
			ivjRegularRateTextField = new  JTextField();
			ivjRegularRateTextField.setName("RegularRateTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjRegularRateTextField;
}

/**
 * Return the JLabel4 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel gettargetRateLabel() {
	if (ivjtargetRateLabel == null) {
		try {
			ivjtargetRateLabel = new  JLabel();
			ivjtargetRateLabel.setName("targetRateLabel");
			ivjtargetRateLabel.setText("Target Rate");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjtargetRateLabel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("DefaultRatePanel");
		setBorder(new EtchedBorder());
		setLayout(new  GridBagLayout());
		setSize(388, 215);

		GridBagConstraints constraintstargetRateLabel = new  GridBagConstraints();
		constraintstargetRateLabel.gridx = 1; constraintstargetRateLabel.gridy = 1;
		constraintstargetRateLabel.insets = new  Insets(23, 50, 8, 65);
		add(gettargetRateLabel(), constraintstargetRateLabel);

		GridBagConstraints constraintsRegularRateTextField = new  GridBagConstraints();
		constraintsRegularRateTextField.gridx = 2; constraintsRegularRateTextField.gridy =
		1;
		constraintsRegularRateTextField.fill =  GridBagConstraints.HORIZONTAL;
		constraintsRegularRateTextField.weightx = 1.0;
		constraintsRegularRateTextField.ipadx = 60;
		constraintsRegularRateTextField.insets = new  Insets(18, 11, 9, 132);
		add(getRegularRateTextField(), constraintsRegularRateTextField);

		GridBagConstraints constraintsOvertimeRateLabel = new  GridBagConstraints();
		constraintsOvertimeRateLabel.gridx = 1; constraintsOvertimeRateLabel.gridy = 2;
		constraintsOvertimeRateLabel.insets = new  Insets(15, 50, 8, 10);
		add(getOvertimeRateLabel(), constraintsOvertimeRateLabel);

		GridBagConstraints constraintsOvertimeRateTextField = new  GridBagConstraints();
		constraintsOvertimeRateTextField.gridx = 2; constraintsOvertimeRateTextField.gridy
		= 2;
		constraintsOvertimeRateTextField.fill =  GridBagConstraints.HORIZONTAL;
		constraintsOvertimeRateTextField.weightx = 1.0;
		constraintsOvertimeRateTextField.ipadx = 60;
		constraintsOvertimeRateTextField.insets = new  Insets(9, 11, 10, 132);
		add(getOvertimeRateTextField(), constraintsOvertimeRateTextField);

		GridBagConstraints constraintsRegularCostLabel = new  GridBagConstraints();
		constraintsRegularCostLabel.gridx = 1; constraintsRegularCostLabel.gridy = 3;
		constraintsRegularCostLabel.insets = new  Insets(15, 50, 7, 58);
		add(getRegularCostLabel(), constraintsRegularCostLabel);

		GridBagConstraints constraintsRegularCostTextField = new  GridBagConstraints();
		constraintsRegularCostTextField.gridx = 2; constraintsRegularCostTextField.gridy =
		3;
		constraintsRegularCostTextField.fill =  GridBagConstraints.HORIZONTAL;
		constraintsRegularCostTextField.weightx = 1.0;
		constraintsRegularCostTextField.ipadx = 60;
		constraintsRegularCostTextField.insets = new  Insets(8, 11, 10, 132);
		add(getRegularCostTextField(), constraintsRegularCostTextField);

		GridBagConstraints constraintsOvertimeCostLabel = new  GridBagConstraints();
		constraintsOvertimeCostLabel.gridx = 1; constraintsOvertimeCostLabel.gridy = 4;
		constraintsOvertimeCostLabel.insets = new  Insets(16, 50, 7, 50);
		add(getOvertimeCostLabel(), constraintsOvertimeCostLabel);

		GridBagConstraints constraintsOvertimeCostTextField = new  GridBagConstraints();
		constraintsOvertimeCostTextField.gridx = 2; constraintsOvertimeCostTextField.gridy
		= 4;
		constraintsOvertimeCostTextField.fill =  GridBagConstraints.HORIZONTAL;
		constraintsOvertimeCostTextField.weightx = 1.0;
		constraintsOvertimeCostTextField.ipadx = 60;
		constraintsOvertimeCostTextField.insets = new  Insets(8, 11, 11, 132);
		add(getOvertimeCostTextField(), constraintsOvertimeCostTextField);

		GridBagConstraints constraintsOfficeLabel = new  GridBagConstraints();
		constraintsOfficeLabel.gridx = 1; constraintsOfficeLabel.gridy = 5;
		constraintsOfficeLabel.insets = new  Insets(16, 50, 25, 97);
		add(getOfficeLabel(), constraintsOfficeLabel);

		GridBagConstraints constraintsOfficeComboBox = new  GridBagConstraints();
		constraintsOfficeComboBox.gridx = 2; constraintsOfficeComboBox.gridy = 5;
		constraintsOfficeComboBox.fill =  GridBagConstraints.HORIZONTAL;
		constraintsOfficeComboBox.weightx = 1.0;
		constraintsOfficeComboBox.ipadx = -44;
		constraintsOfficeComboBox.insets = new  Insets(7, 11, 22, 114);
		add(getOfficeComboBox(), constraintsOfficeComboBox);
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		 JFrame frame = new  JFrame();
		DefaultRatePanel aDefaultRatePanel;
```

```
			aDefaultRatePanel = new DefaultRatePanel();
			frame.setContentPane(aDefaultRatePanel);
			frame.setSize(aDefaultRatePanel.getSize());
			frame.addWindowListener(new  WindowAdapter() {
				public void windowClosing( WindowEvent e) {
					System.exit(0);
				};
			});
			frame.setVisible(true);
		} catch (Throwable exception) {
			System.err.println("Exception occurred in main() of  JPanel");
			exception.printStackTrace(System.out);
		}
}
}


/*
 * Revision History
 *
 * 1/29/00 Eliminated the target rate label and text field
 *
 */
```

LoginDialog.java

```
//------------------------------------------------------------------------------
// LoginDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/19/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//------------------------------------------------------------------------------

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

/**
 * A dialog class that is used for loggin in to the program.
 * It creates the necessary form that will be able to obtain all the necessary
 * information in order to calidate the user.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/
class LoginDialog extends  JDialog {
// Private data members
	private  JPanel ivjinputPanel = null;
	private  JPanel ivjJDialogContentPane = null;
	private  JPasswordField ivjLoginPasswordField = null;
	private ModifyButtonsPanel ivjModifyButtonsPanel1 = null;
	private  JLabel ivjNameLabel = null;
	private  JTextField ivjNameTextField = null;
	private  JLabel ivjPasswordLabel = null;

/**
 * LoginDialog constructor comment.
 */
```

```
public LoginDialog() {
	super();
	initialize();
}

/**
 * LoginDialog constructor comment.
 * @param owner  Frame
 */
public LoginDialog( Frame owner) {
	super(owner);
}

/**
 * LoginDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public LoginDialog( Frame owner, String title) {
	super(owner, title);
}

/**
 * LoginDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public LoginDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}

/**
 * LoginDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public LoginDialog( Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the inputPanel property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getinputPanel() {
	if (ivjinputPanel == null) {
		try {
			ivjinputPanel = new  JPanel();
			ivjinputPanel.setName("inputPanel");
			ivjinputPanel.setLayout(new  GridBagLayout());
			ivjinputPanel.setBounds(30, 27, 328, 80);

			GridBagConstraints constraintsNameTextField = new  GridBagConstraints();
			constraintsNameTextField.gridx = 2; constraintsNameTextField.gridy = 1;
			constraintsNameTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsNameTextField.weightx = 1.0;
			constraintsNameTextField.ipadx = 174;
			constraintsNameTextField.insets = new  Insets(14, 2, 6, 48);
			getinputPanel().add(getNameTextField(), constraintsNameTextField);

			GridBagConstraints constraintsPasswordLabel = new  GridBagConstraints();
			constraintsPasswordLabel.gridx = 1; constraintsPasswordLabel.gridy = 2;
			constraintsPasswordLabel.ipadx = 23;
			constraintsPasswordLabel.insets = new  Insets(8, 18, 18, 1);
			getinputPanel().add(getPasswordLabel(), constraintsPasswordLabel);

			GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
			constraintsNameLabel.gridx = 1; constraintsNameLabel.gridy = 1;
			constraintsNameLabel.ipadx = 12;
			constraintsNameLabel.insets = new  Insets(16, 18, 8, 37);
			getinputPanel().add(getNameLabel(), constraintsNameLabel);

			GridBagConstraints constraintsLoginPasswordField = new
			GridBagConstraints();
			constraintsLoginPasswordField.gridx = 2;
			constraintsLoginPasswordField.gridy = 2;
			constraintsLoginPasswordField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsLoginPasswordField.weightx = 1.0;
			constraintsLoginPasswordField.ipadx = 174;
			constraintsLoginPasswordField.insets = new  Insets(6, 2, 16, 48);
			getinputPanel().add(getLoginPasswordField(),
			constraintsLoginPasswordField);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjinputPanel;
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getinputPanel(), getinputPanel().getName());
			getJDialogContentPane().add(getModifyButtonsPanel1(),
			getModifyButtonsPanel1().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the LoginPasswordField property value.
 * @return  JPasswordField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPasswordField getLoginPasswordField() {
	if (ivjLoginPasswordField == null) {
		try {
			ivjLoginPasswordField = new  JPasswordField();
			ivjLoginPasswordField.setName("LoginPasswordField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjLoginPasswordField;
}

/**
```

```
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel1() {
	if (ivjModifyButtonsPanel1 == null) {
		try {
			ivjModifyButtonsPanel1 = new ModifyButtonsPanel();
			ivjModifyButtonsPanel1.setName("ModifyButtonsPanel1");
			ivjModifyButtonsPanel1.setBounds(78, 134, 242, 56);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyButtonsPanel1;
}

/**
 * Return the NameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
	if (ivjNameLabel == null) {
		try {
			ivjNameLabel = new  JLabel();
			ivjNameLabel.setName("NameLabel");
			ivjNameLabel.setText("Name");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameLabel;
}

/**
 * Return the NameTextField property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
	if (ivjNameTextField == null) {
		try {
			ivjNameTextField = new  JTextField();
			ivjNameTextField.setName("NameTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameTextField;
}

/**
 * Return the PasswordLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getPasswordLabel() {
	if (ivjPasswordLabel == null) {
```

```
                try {
                        ivjPasswordLabel = new  JLabel();
                        ivjPasswordLabel.setName("PasswordLabel");
                        ivjPasswordLabel.setText("Password");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjPasswordLabel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("LoginDialog");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setSize(388, 218);
                setTitle("Login");
                setContentPane(getJDialogContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                LoginDialog aLoginDialog;
                aLoginDialog = new LoginDialog();
                aLoginDialog.setModal(true);
                aLoginDialog.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aLoginDialog.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JDialog");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

MachineFormPanel.java

```
//---------------------------------------------------------------------------
// MachineFormPanel.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified 1/19/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------

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


/**
 * A panel class that is used with information regarding a machine.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for viewing and modifiying a machine resource.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class MachineFormPanel extends  JPanel {
       private DefaultRatePanel ivjDefaultRatePanel = null;
       private  JLabel ivjDefaultsLabel = null;
       private  JLabel ivjDescriptionLabel = null;
       private  JTextArea ivjDescriptionTextArea = null;
       private  JLabel ivjHeadingLabel = null;
       private  JPanel ivjInfoPanel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private  JLabel ivjNameLabel = null;
       private  JTextField ivjNameTextField = null;
       private  JLabel ivjSerialNumberLabel = null;
       private  JTextField ivjSerialNumberTextField = null;

/**
 * MachineFormPanel constructor comment.
 */
public MachineFormPanel() {
       super();
       initialize();
}

/**
 * MachineFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public MachineFormPanel( LayoutManager layout) {
       super(layout);
}

/**
 * MachineFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
```

```
public MachineFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
	super(layout, isDoubleBuffered);
}

/**
 * MachineFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public MachineFormPanel(boolean isDoubleBuffered) {
	super(isDoubleBuffered);
}

/**
 * Return the DefaultRatePanel2 property value.
 * @return DefaultRatePanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private DefaultRatePanel getDefaultRatePanel() {
	if (ivjDefaultRatePanel == null) {
		try {
			ivjDefaultRatePanel = new DefaultRatePanel();
			ivjDefaultRatePanel.setName("DefaultRatePanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjDefaultRatePanel;
}

/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getDefaultsLabel() {
	if (ivjDefaultsLabel == null) {
		try {
			ivjDefaultsLabel = new  JLabel();
			ivjDefaultsLabel.setName("DefaultsLabel");
			ivjDefaultsLabel.setText("Defaults");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjDefaultsLabel;
}

/**
 * Return the JLabel4 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getDescriptionLabel() {
	if (ivjDescriptionLabel == null) {
		try {
			ivjDescriptionLabel = new  JLabel();
			ivjDescriptionLabel.setName("DescriptionLabel");
			ivjDescriptionLabel.setText("Description");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
```

```
                        handleException(ivjExc);
                }
        }
        return ivjDescriptionLabel;
}

/**
 * Return the JTextArea1 property value.
 * @return  JTextArea
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextArea getDescriptionTextArea() {
        if (ivjDescriptionTextArea == null) {
                try {
                        ivjDescriptionTextArea = new  JTextArea();
                        ivjDescriptionTextArea.setName("DescriptionTextArea");
                        ivjDescriptionTextArea.setLineWrap(true);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjDescriptionTextArea;
}

/**
 * Return the JLabel5 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
        if (ivjHeadingLabel == null) {
                try {
                        ivjHeadingLabel = new  JLabel();
                        ivjHeadingLabel.setName("HeadingLabel");
                        ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
                        ivjHeadingLabel.setText("Machine Information");
                        ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHeadingLabel;
}

/**
 * Return the JPanel1 property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getInfoPanel() {
        if (ivjInfoPanel == null) {
                try {
                        ivjInfoPanel = new  JPanel();
                        ivjInfoPanel.setName("InfoPanel");
                        ivjInfoPanel.setLayout(new  GridBagLayout());

                        GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
                        constraintsNameLabel.gridx = 1; constraintsNameLabel.gridy = 1;
                        constraintsNameLabel.ipadx = 12;
                        constraintsNameLabel.insets = new  Insets(23, 30, 11, 56);
                        getInfoPanel().add(getNameLabel(), constraintsNameLabel);
```

```
			GridBagConstraints constraintsSerialNumberLabel = new
			GridBagConstraints();
			constraintsSerialNumberLabel.gridx = 1; constraintsSerialNumberLabel.gridy
			= 2;
			constraintsSerialNumberLabel.ipadx = 16;
			constraintsSerialNumberLabel.insets = new  Insets(12, 30, 11, 4);
			                      getInfoPanel().add(getSerialNumberLabel(),
			constraintsSerialNumberLabel);

			GridBagConstraints constraintsNameTextField = new  GridBagConstraints();
			constraintsNameTextField.gridx = 2; constraintsNameTextField.gridy = 1;
			constraintsNameTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsNameTextField.weightx = 1.0;
			constraintsNameTextField.ipadx = 215;
			constraintsNameTextField.insets = new  Insets(21, 5, 9, 61);
			getInfoPanel().add(getNameTextField(), constraintsNameTextField);

			GridBagConstraints constraintsSerialNumberTextField = new
			GridBagConstraints();
			constraintsSerialNumberTextField.gridx = 2;
			constraintsSerialNumberTextField.gridy = 2;
			constraintsSerialNumberTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsSerialNumberTextField.weightx = 1.0;
			constraintsSerialNumberTextField.ipadx = 215;
			constraintsSerialNumberTextField.insets = new  Insets(10, 5, 9, 61);
			getInfoPanel().add(getSerialNumberTextField(),
			constraintsSerialNumberTextField);

			GridBagConstraints constraintsDescriptionLabel = new
			GridBagConstraints();
			constraintsDescriptionLabel.gridx = 1; constraintsDescriptionLabel.gridy =
			3;
			constraintsDescriptionLabel.insets = new  Insets(33, 30, 48, 36);
			getInfoPanel().add(getDescriptionLabel(), constraintsDescriptionLabel);

			GridBagConstraints constraintsDescriptionTextArea = new
			GridBagConstraints();
			constraintsDescriptionTextArea.gridx = 2;
			constraintsDescriptionTextArea.gridy = 3;
			constraintsDescriptionTextArea.fill =  GridBagConstraints.BOTH;
			constraintsDescriptionTextArea.weightx = 1.0;
			constraintsDescriptionTextArea.weighty = 1.0;
			constraintsDescriptionTextArea.ipady = 44;
			constraintsDescriptionTextArea.insets = new  Insets(9, 4, 24, 62);
			getInfoPanel().add(getDescriptionTextArea(),
			constraintsDescriptionTextArea);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjInfoPanel;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
	if (ivjModifyButtonsPanel == null) {
		try {
			ivjModifyButtonsPanel = new ModifyButtonsPanel();
			ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
```

```
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjModifyButtonsPanel;
}

/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
		if (ivjNameLabel == null) {
			try {
				ivjNameLabel = new  JLabel();
				ivjNameLabel.setName("NameLabel");
				ivjNameLabel.setText("Name");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjNameLabel;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
		if (ivjNameTextField == null) {
			try {
				ivjNameTextField = new  JTextField();
				ivjNameTextField.setName("NameTextField");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjNameTextField;
}

/**
 * Return the JLabel3 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getSerialNumberLabel() {
		if (ivjSerialNumberLabel == null) {
			try {
				ivjSerialNumberLabel = new  JLabel();
				ivjSerialNumberLabel.setName("SerialNumberLabel");
				ivjSerialNumberLabel.setText("Serial Number");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjSerialNumberLabel;
}
```

```
/**
 * Return the JTextField11 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getSerialNumberTextField() {
	if (ivjSerialNumberTextField == null) {
		try {
			ivjSerialNumberTextField = new  JTextField();
			ivjSerialNumberTextField.setName("SerialNumberTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjSerialNumberTextField;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("MachineFormPanel");
		setLayout(new  GridBagLayout());
		setSize(496, 624);

		GridBagConstraints constraintsDefaultsLabel = new  GridBagConstraints();
		constraintsDefaultsLabel.gridx = 1; constraintsDefaultsLabel.gridy = 3;
		constraintsDefaultsLabel.ipadx = 14;
		constraintsDefaultsLabel.insets = new  Insets(8, 40, 1, 13);
		add(getDefaultsLabel(), constraintsDefaultsLabel);

		GridBagConstraints constraintsDefaultRatePanel = new  GridBagConstraints();
		constraintsDefaultRatePanel.gridx = 1; constraintsDefaultRatePanel.gridy = 4;
		constraintsDefaultRatePanel.gridwidth = 2;
		constraintsDefaultRatePanel.fill =  GridBagConstraints.BOTH;
		constraintsDefaultRatePanel.weightx = 1.0;
		constraintsDefaultRatePanel.weighty = 1.0;
		constraintsDefaultRatePanel.ipadx = -4;
		constraintsDefaultRatePanel.ipady = 11;
		constraintsDefaultRatePanel.insets = new  Insets(2, 40, 7, 68);
		add(getDefaultRatePanel(), constraintsDefaultRatePanel);

		GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
		constraintsHeadingLabel.gridx = 1; constraintsHeadingLabel.gridy = 1;
		constraintsHeadingLabel.gridwidth = 2;
		constraintsHeadingLabel.ipadx = 94;
		constraintsHeadingLabel.ipady = 20;
		constraintsHeadingLabel.insets = new  Insets(25, 82, 2, 83);
		add(getHeadingLabel(), constraintsHeadingLabel);

		GridBagConstraints constraintsInfoPanel = new  GridBagConstraints();
		constraintsInfoPanel.gridx = 1; constraintsInfoPanel.gridy = 2;
		constraintsInfoPanel.gridwidth = 2;
		constraintsInfoPanel.fill =  GridBagConstraints.BOTH;
		constraintsInfoPanel.weightx = 1.0;
		constraintsInfoPanel.weighty = 1.0;
		constraintsInfoPanel.ipady = 15;
		constraintsInfoPanel.insets = new  Insets(3, 40, 7, 40);
		add(getInfoPanel(), constraintsInfoPanel);

		GridBagConstraints constraintsModifyButtonsPanel = new  GridBagConstraints();
		constraintsModifyButtonsPanel.gridx = 2; constraintsModifyButtonsPanel.gridy = 5;
		constraintsModifyButtonsPanel.fill =  GridBagConstraints.BOTH;
		constraintsModifyButtonsPanel.weightx = 1.0;
		constraintsModifyButtonsPanel.weighty = 1.0;
		constraintsModifyButtonsPanel.insets = new  Insets(8, 13, 14, 127);
		add(getModifyButtonsPanel(), constraintsModifyButtonsPanel);
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		 JFrame frame = new  JFrame();
		MachineFormPanel aMachineFormPanel;
		aMachineFormPanel = new MachineFormPanel();
		frame.setContentPane(aMachineFormPanel);
		frame.setSize(aMachineFormPanel.getSize());
		frame.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		frame.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JPanel");
		exception.printStackTrace(System.out);
	}
}
}


/*
 * Revision History
 *
 */
```

MainFrame.java

```
//-----------------------------------------------------------------------------
// MainFrame.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/17/00
// Last Modified 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
```

```
//------------------------------------------------------------------------------

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


/**
 * A frame class that is used for the main display in the program.
 * It arranges all the information to be displayed into regions.  The regions
 * contain trees, graphs and the panels that hold the objects information.
 *
 * @version   1.1
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class MainFrame extends  JFrame {
// Private data members
       private  JPanel ivjJFrameContentPane = null;
       private  JMenuItem ivjAddClientMenuItem = null;
       private  JMenuItem ivjAddConsultantMenuItem = null;
       private  JMenuItem ivjAddMachineMenuItem = null;
       private  JMenuItem ivjAddOfficeMenuItem = null;
       private  JMenuItem ivjAddProjectMenuItem = null;
       private  JMenuItem ivjAddResourceAllocationMenuItem = null;
       private  JMenu ivjAddResourceMenu = null;
       private  JMenuItem ivjAddRoomMenuItem = null;
       private  JMenuItem ivjAddUserMenuItem = null;
       private  JMenu ivjAdministratorMenu = null;
       private  JMenuItem ivjChangePasswordMenuItem = null;
       private  JScrollPane ivjEquipmentViewTab = null;
       private  JMenuItem ivjExitMenuItem = null;
       private  JMenu ivjFileMenu = null;
       private  JTabbedPane ivjGraphSelectTabbedPane = null;
       private  JMenu ivjHelpMenu = null;
       private  JScrollPane ivjInfoPane = null;
       private  JMenuItem ivjModifyUserMenuItem = null;
       private  JMenu ivjProjectLeaderMenu = null;
       private  JScrollPane ivjProjectViewTab = null;
       private  JScrollPane ivjRoomViewTab = null;
       private  JScrollPane ivjStaffViewTab = null;
       private  JMenuItem ivjViewGraphMenuItem = null;
       private  JMenu ivjViewMenu = null;
       private  JMenuItem ivjViewUserMenuItem = null;
       private  JScrollPane ivjProjectScrollPane = null;
       private  JTree ivjProjectTree = null;
       private  JTree ivjResourceTree = null;
       private  JScrollPane ivjTreeScrollPane = null;
       private  JMenuItem ivjAboutMenuItem = null;
       private  JMenuItem ivjaddTimeslotMenuItem = null;
       private  JMenuItem ivjDeleteTimeslotMenuItem = null;
       private  JLabel ivjEndDateLabel = null;
       private  JTextField ivjEndDateTextField = null;
       private  JLabel ivjStartDateLabel = null;
       private  JTextField ivjStartDateTextField = null;
       private  JTabbedPane ivjTreeSelectTabbedPane = null;
       private  JPanel ivjGraphDatePanel = null;
       private  JMenuBar ivjMainFrameJMenuBar = null;

/**
 * MainGUI constructor comment.
 */
public MainFrame() {
       super();
       initialize();
}

/**
 * MainGUI constructor comment.
 * @param title  String
 */
public MainFrame(String title) {
	super(title);
}

/**
 * Return the AboutMenuItem property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAboutMenuItem() {
	if (ivjAboutMenuItem == null) {
		try {
			ivjAboutMenuItem = new  JMenuItem();
			ivjAboutMenuItem.setName("AboutMenuItem");
			ivjAboutMenuItem.setText("About");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAboutMenuItem;
}

/**
 * Return the JMenuItem13 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddClientMenuItem() {
	if (ivjAddClientMenuItem == null) {
		try {
			ivjAddClientMenuItem = new  JMenuItem();
			ivjAddClientMenuItem.setName("AddClientMenuItem");
			ivjAddClientMenuItem.setText("Add Client");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAddClientMenuItem;
}

/**
 * Return the JMenuItem10 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddConsultantMenuItem() {
	if (ivjAddConsultantMenuItem == null) {
		try {
			ivjAddConsultantMenuItem = new  JMenuItem();
			ivjAddConsultantMenuItem.setName("AddConsultantMenuItem");
			ivjAddConsultantMenuItem.setText("Add Consultant");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAddConsultantMenuItem;
}
```

```
/**
 * Return the JMenuItem9 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddMachineMenuItem() {
        if (ivjAddMachineMenuItem == null) {
                try {
                        ivjAddMachineMenuItem = new  JMenuItem();
                        ivjAddMachineMenuItem.setName("AddMachineMenuItem");
                        ivjAddMachineMenuItem.setText("Add Machine");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAddMachineMenuItem;
}

/**
 * Return the JMenuItem7 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddOfficeMenuItem() {
        if (ivjAddOfficeMenuItem == null) {
                try {
                        ivjAddOfficeMenuItem = new  JMenuItem();
                        ivjAddOfficeMenuItem.setName("AddOfficeMenuItem");
                        ivjAddOfficeMenuItem.setText("Add Office");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAddOfficeMenuItem;
}

/**
 * Return the AddProjectMenuItem property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddProjectMenuItem() {
        if (ivjAddProjectMenuItem == null) {
                try {
                        ivjAddProjectMenuItem = new  JMenuItem();
                        ivjAddProjectMenuItem.setName("AddProjectMenuItem");
                        ivjAddProjectMenuItem.setText("Add Project");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAddProjectMenuItem;
}

/**
 * Return the AddResourceAllocationMenuItem property value.
 * @return  JMenuItem
 */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddResourceAllocationMenuItem() {
	if (ivjAddResourceAllocationMenuItem == null) {
		try {
			ivjAddResourceAllocationMenuItem = new  JMenuItem();
			ivjAddResourceAllocationMenuItem.setName("AddResourceAllocationMenuItem");
			ivjAddResourceAllocationMenuItem.setText("Add Resource Allocation");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAddResourceAllocationMenuItem;
}

/**
 * Return the JMenu5 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getAddResourceMenu() {
	if (ivjAddResourceMenu == null) {
		try {
			ivjAddResourceMenu = new  JMenu();
			ivjAddResourceMenu.setName("AddResourceMenu");
			ivjAddResourceMenu.setText("Add Resource");
			ivjAddResourceMenu.add(getAddConsultantMenuItem());
			ivjAddResourceMenu.add(getAddMachineMenuItem());
			ivjAddResourceMenu.add(getAddRoomMenuItem());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAddResourceMenu;
}

/**
 * Return the JMenuItem8 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddRoomMenuItem() {
	if (ivjAddRoomMenuItem == null) {
		try {
			ivjAddRoomMenuItem = new  JMenuItem();
			ivjAddRoomMenuItem.setName("AddRoomMenuItem");
			ivjAddRoomMenuItem.setText("Add Room");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAddRoomMenuItem;
}

/**
 * Return the addTimeslotMenuItem property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getaddTimeslotMenuItem() {
        if (ivjaddTimeslotMenuItem == null) {
                try {
                        ivjaddTimeslotMenuItem = new  JMenuItem();
                        ivjaddTimeslotMenuItem.setName("addTimeslotMenuItem");
                        ivjaddTimeslotMenuItem.setText("Add Timeslot");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjaddTimeslotMenuItem;
}

/**
 * Return the JMenuItem6 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getAddUserMenuItem() {
        if (ivjAddUserMenuItem == null) {
                try {
                        ivjAddUserMenuItem = new  JMenuItem();
                        ivjAddUserMenuItem.setName("AddUserMenuItem");
                        ivjAddUserMenuItem.setText("Add User");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAddUserMenuItem;
}

/**
 * Return the JMenu4 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getAdministratorMenu() {
        if (ivjAdministratorMenu == null) {
                try {
                        ivjAdministratorMenu = new  JMenu();
                        ivjAdministratorMenu.setName("AdministratorMenu");
                        ivjAdministratorMenu.setText("Administrator");
                        ivjAdministratorMenu.add(getAddClientMenuItem());
                        ivjAdministratorMenu.add(getAddProjectMenuItem());
                        ivjAdministratorMenu.add(getAddResourceMenu());
                        ivjAdministratorMenu.add(getAddOfficeMenuItem());
                        ivjAdministratorMenu.add(getAddResourceAllocationMenuItem());
                        ivjAdministratorMenu.add(getAddUserMenuItem());
                        ivjAdministratorMenu.add(getModifyUserMenuItem());
                        ivjAdministratorMenu.add(getViewUserMenuItem());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAdministratorMenu;
}

/**
 * Return the JMenuItem2 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getChangePasswordMenuItem() {
	if (ivjChangePasswordMenuItem == null) {
		try {
			ivjChangePasswordMenuItem = new  JMenuItem();
			ivjChangePasswordMenuItem.setName("ChangePasswordMenuItem");
			ivjChangePasswordMenuItem.setText("Change Password");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjChangePasswordMenuItem;
}

/**
 * Return the DeleteTimeslotMenuItem property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getDeleteTimeslotMenuItem() {
	if (ivjDeleteTimeslotMenuItem == null) {
		try {
			ivjDeleteTimeslotMenuItem = new  JMenuItem();
			ivjDeleteTimeslotMenuItem.setName("DeleteTimeslotMenuItem");
			ivjDeleteTimeslotMenuItem.setText("Delete Timeslot");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjDeleteTimeslotMenuItem;
}

/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getEndDateLabel() {
	if (ivjEndDateLabel == null) {
		try {
			ivjEndDateLabel = new  JLabel();
			ivjEndDateLabel.setName("EndDateLabel");
			ivjEndDateLabel.setText("End Date");
			ivjEndDateLabel.setForeground( Color.black);
			ivjEndDateLabel.setHorizontalTextPosition( SwingConstants.LEFT);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjEndDateLabel;
}

/**
 * Return the JTextField2 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getEndDateTextField() {
        if (ivjEndDateTextField == null) {
                try {
                        ivjEndDateTextField = new  JTextField();
                        ivjEndDateTextField.setName("EndDateTextField");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjEndDateTextField;
}

/**
 * Return the Tab3 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getEquipmentViewTab() {
        if (ivjEquipmentViewTab == null) {
                try {
                        ivjEquipmentViewTab = new  JScrollPane();
                        ivjEquipmentViewTab.setName("EquipmentViewTab");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjEquipmentViewTab;
}

/**
 * Return the JMenuItem1 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getExitMenuItem() {
        if (ivjExitMenuItem == null) {
                try {
                        ivjExitMenuItem = new  JMenuItem();
                        ivjExitMenuItem.setName("ExitMenuItem");
                        ivjExitMenuItem.setText("Exit");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjExitMenuItem;
}

/**
 * Return the JMenu1 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getFileMenu() {
        if (ivjFileMenu == null) {
                try {
                        ivjFileMenu = new  JMenu();
                        ivjFileMenu.setName("FileMenu");
                        ivjFileMenu.setText("File");
                        ivjFileMenu.add(getChangePasswordMenuItem());
                        ivjFileMenu.add(getExitMenuItem());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjFileMenu;
}

/**
 * Return the JPanel1 property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getGraphDatePanel() {
        if (ivjGraphDatePanel == null) {
                try {
                        ivjGraphDatePanel = new  JPanel();
                        ivjGraphDatePanel.setName("GraphDatePanel");
                        ivjGraphDatePanel.setLayout(new  GridBagLayout());

                        GridBagConstraints constraintsStartDateLabel = new  GridBagConstraints();
                        constraintsStartDateLabel.gridx = 1; constraintsStartDateLabel.gridy = 1;
                        constraintsStartDateLabel.ipadx = 4;
                        constraintsStartDateLabel.insets = new  Insets(7, 10, 7, 2);
                        getGraphDatePanel().add(getStartDateLabel(), constraintsStartDateLabel);

                        GridBagConstraints constraintsStartDateTextField = new
                        GridBagConstraints();
                        constraintsStartDateTextField.gridx = 2;
                        constraintsStartDateTextField.gridy = 1;
                        constraintsStartDateTextField.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsStartDateTextField.weightx = 1.0;
                        constraintsStartDateTextField.ipadx = 64;
                        constraintsStartDateTextField.insets = new  Insets(5, 2, 5, 9);
                        getGraphDatePanel().add(getStartDateTextField(),
                        constraintsStartDateTextField);

                        GridBagConstraints constraintsEndDateLabel = new  GridBagConstraints();
                        constraintsEndDateLabel.gridx = 3; constraintsEndDateLabel.gridy = 1;
                        constraintsEndDateLabel.ipadx = 1;
                        constraintsEndDateLabel.insets = new  Insets(7, 9, 7, 2);
                        getGraphDatePanel().add(getEndDateLabel(), constraintsEndDateLabel);

                        GridBagConstraints constraintsEndDateTextField = new
                        GridBagConstraints();
                        constraintsEndDateTextField.gridx = 4; constraintsEndDateTextField.gridy =
                        1;
                        constraintsEndDateTextField.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsEndDateTextField.weightx = 1.0;
                        constraintsEndDateTextField.ipadx = 64;
                        constraintsEndDateTextField.insets = new  Insets(5, 3, 5, 309);
                        getGraphDatePanel().add(getEndDateTextField(),
                        constraintsEndDateTextField);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjGraphDatePanel;
}

/**
 * Return the JTabbedPane1 property value.
 * @return  JTabbedPane
 */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTabbedPane getGraphSelectTabbedPane() {
	if (ivjGraphSelectTabbedPane == null) {
		try {
			ivjGraphSelectTabbedPane = new  JTabbedPane();
			ivjGraphSelectTabbedPane.setName("GraphSelectTabbedPane");
			ivjGraphSelectTabbedPane.setTabPlacement( JTabbedPane.TOP);
			ivjGraphSelectTabbedPane.insertTab("Project View", null,
			getProjectViewTab(), null, 0);
			ivjGraphSelectTabbedPane.insertTab("Staff View", null, getStaffViewTab(),
			null, 1);
			ivjGraphSelectTabbedPane.insertTab("Equipment View", null,
			getEquipmentViewTab(), null, 2);
			ivjGraphSelectTabbedPane.insertTab("Room View", null, getRoomViewTab(),
			null, 3);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjGraphSelectTabbedPane;
}

/**
 * Return the JMenu3 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getHelpMenu() {
	if (ivjHelpMenu == null) {
		try {
			ivjHelpMenu = new  JMenu();
			ivjHelpMenu.setName("HelpMenu");
			ivjHelpMenu.setText("Help");
			ivjHelpMenu.add(getAboutMenuItem());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjHelpMenu;
}

/**
 * Return the JScrollPane2 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getInfoPane() {
	if (ivjInfoPane == null) {
		try {
			ivjInfoPane = new  JScrollPane();
			ivjInfoPane.setName("InfoPane");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjInfoPane;
}

/**
```

```
 * Return the JFrameContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJFrameContentPane() {
	if (ivjJFrameContentPane == null) {
		try {
			ivjJFrameContentPane = new  JPanel();
			ivjJFrameContentPane.setName("JFrameContentPane");
			ivjJFrameContentPane.setLayout(new  GridBagLayout());
			ivjJFrameContentPane.setVisible(true);

			GridBagConstraints constraintsGraphSelectTabbedPane = new
			GridBagConstraints();
			constraintsGraphSelectTabbedPane.gridx = 2;
			constraintsGraphSelectTabbedPane.gridy = 2;
			constraintsGraphSelectTabbedPane.fill =  GridBagConstraints.BOTH;
			constraintsGraphSelectTabbedPane.weightx = 1.0;
			constraintsGraphSelectTabbedPane.weighty = 1.0;
			constraintsGraphSelectTabbedPane.ipadx = 646;
			constraintsGraphSelectTabbedPane.ipady = 201;
			constraintsGraphSelectTabbedPane.insets = new  Insets(0, 6, 7, 20);
			getJFrameContentPane().add(getGraphSelectTabbedPane(),
			constraintsGraphSelectTabbedPane);

			GridBagConstraints constraintsInfoPane = new  GridBagConstraints();
			constraintsInfoPane.gridx = 2; constraintsInfoPane.gridy = 3;
			constraintsInfoPane.fill =  GridBagConstraints.BOTH;
			constraintsInfoPane.weightx = 1.0;
			constraintsInfoPane.weighty = 1.0;
			constraintsInfoPane.ipadx = 651;
			constraintsInfoPane.ipady = 267;
			constraintsInfoPane.insets = new  Insets(7, 10, 8, 16);
			getJFrameContentPane().add(getInfoPane(), constraintsInfoPane);

			GridBagConstraints constraintsTreeSelectTabbedPane = new
			GridBagConstraints();
			constraintsTreeSelectTabbedPane.gridx = 1;
			constraintsTreeSelectTabbedPane.gridy = 1;
			constraintsTreeSelectTabbedPane.gridheight = 3;
			constraintsTreeSelectTabbedPane.fill =  GridBagConstraints.BOTH;
			constraintsTreeSelectTabbedPane.weightx = 1.0;
			constraintsTreeSelectTabbedPane.weighty = 1.0;
			constraintsTreeSelectTabbedPane.ipadx = 61;
			constraintsTreeSelectTabbedPane.ipady = 175;
			constraintsTreeSelectTabbedPane.insets = new  Insets(6, 7, 10, 5);
			getJFrameContentPane().add(getTreeSelectTabbedPane(),
			constraintsTreeSelectTabbedPane);

			GridBagConstraints constraintsGraphDatePanel = new  GridBagConstraints();
			constraintsGraphDatePanel.gridx = 2; constraintsGraphDatePanel.gridy = 1;
			constraintsGraphDatePanel.fill =  GridBagConstraints.BOTH;
			constraintsGraphDatePanel.weightx = 1.0;
			constraintsGraphDatePanel.weighty = 1.0;
			constraintsGraphDatePanel.insets = new  Insets(4, 6, 0, 80);
			getJFrameContentPane().add(getGraphDatePanel(),
			constraintsGraphDatePanel);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJFrameContentPane;
}

/**
 * Return the MainFrameJMenuBar property value.
 * @return  JMenuBar
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuBar getMainFrameJMenuBar() {
	if (ivjMainFrameJMenuBar == null) {
		try {
			ivjMainFrameJMenuBar = new  JMenuBar();
			ivjMainFrameJMenuBar.setName("MainFrameJMenuBar");
			ivjMainFrameJMenuBar.add(getFileMenu());
			ivjMainFrameJMenuBar.add(getViewMenu());
			ivjMainFrameJMenuBar.add(getProjectLeaderMenu());
			ivjMainFrameJMenuBar.add(getAdministratorMenu());
			ivjMainFrameJMenuBar.add(getHelpMenu());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjMainFrameJMenuBar;
}

/**
 * Return the JMenuItem5 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getModifyUserMenuItem() {
	if (ivjModifyUserMenuItem == null) {
		try {
			ivjModifyUserMenuItem = new  JMenuItem();
			ivjModifyUserMenuItem.setName("ModifyUserMenuItem");
			ivjModifyUserMenuItem.setText("Modify User");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyUserMenuItem;
}

/**
 * Return the JMenu6 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getProjectLeaderMenu() {
	if (ivjProjectLeaderMenu == null) {
		try {
			ivjProjectLeaderMenu = new  JMenu();
			ivjProjectLeaderMenu.setName("ProjectLeaderMenu");
			ivjProjectLeaderMenu.setText("Project Leader");
			ivjProjectLeaderMenu.add(getaddTimeslotMenuItem());
			ivjProjectLeaderMenu.add(getDeleteTimeslotMenuItem());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjProjectLeaderMenu;
}

/**
 * Return the JScrollPane1 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getProjectScrollPane() {
	if (ivjProjectScrollPane == null) {
		try {
			ivjProjectScrollPane = new  JScrollPane();
			ivjProjectScrollPane.setName("ProjectScrollPane");
			getProjectScrollPane().setViewportView(getProjectTree());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjProjectScrollPane;
}

/**
 * Return the JTree1 property value.
 * @return  JTree
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTree getProjectTree() {
	if (ivjProjectTree == null) {
		try {
			ivjProjectTree = new  JTree();
			ivjProjectTree.setName("ProjectTree");
			ivjProjectTree.setBounds(0, 0, 160, 120);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjProjectTree;
}

/**
 * Return the JScrollPane3 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getProjectViewTab() {
	if (ivjProjectViewTab == null) {
		try {
			ivjProjectViewTab = new  JScrollPane();
			ivjProjectViewTab.setName("ProjectViewTab");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjProjectViewTab;
}

/**
 * Return the JTree2 property value.
 * @return  JTree
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTree getResourceTree() {
	if (ivjResourceTree == null) {
		try {
```

```
                        ivjResourceTree = new  JTree();
                        ivjResourceTree.setName("ResourceTree");
                        ivjResourceTree.setBounds(0, 0, 160, 120);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjResourceTree;
}

/**
 * Return the Tab4 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getRoomViewTab() {
        if (ivjRoomViewTab == null) {
                try {
                        ivjRoomViewTab = new  JScrollPane();
                        ivjRoomViewTab.setName("RoomViewTab");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjRoomViewTab;
}

/**
 * Return the Tab2 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getStaffViewTab() {
        if (ivjStaffViewTab == null) {
                try {
                        ivjStaffViewTab = new  JScrollPane();
                        ivjStaffViewTab.setName("StaffViewTab");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjStaffViewTab;
}

/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getStartDateLabel() {
        if (ivjStartDateLabel == null) {
                try {
                        ivjStartDateLabel = new  JLabel();
                        ivjStartDateLabel.setName("StartDateLabel");
                        ivjStartDateLabel.setText("Start Date");
                        ivjStartDateLabel.setForeground( Color.black);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjStartDateLabel;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getStartDateTextField() {
        if (ivjStartDateTextField == null) {
                try {
                        ivjStartDateTextField = new  JTextField();
                        ivjStartDateTextField.setName("StartDateTextField");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjStartDateTextField;
}

/**
 * Return the JScrollPane2 property value.
 * @return  JScrollPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JScrollPane getTreeScrollPane() {
        if (ivjTreeScrollPane == null) {
                try {
                        ivjTreeScrollPane = new  JScrollPane();
                        ivjTreeScrollPane.setName("TreeScrollPane");
                        getTreeScrollPane().setViewportView(getResourceTree());
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjTreeScrollPane;
}

/**
 * Return the JTabbedPane1 property value.
 * @return  JTabbedPane
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTabbedPane getTreeSelectTabbedPane() {
        if (ivjTreeSelectTabbedPane == null) {
                try {
                        ivjTreeSelectTabbedPane = new  JTabbedPane();
                        ivjTreeSelectTabbedPane.setName("TreeSelectTabbedPane");
                        ivjTreeSelectTabbedPane.insertTab("Project", null, getProjectScrollPane(),
                        null, 0);
                        ivjTreeSelectTabbedPane.insertTab("Resource", null, getTreeScrollPane(),
                        null, 1);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
```

```
				}
		}
		return ivjTreeSelectTabbedPane;
}

/**
 * Return the JMenuItem3 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getViewGraphMenuItem() {
		if (ivjViewGraphMenuItem == null) {
				try {
						ivjViewGraphMenuItem = new  JMenuItem();
						ivjViewGraphMenuItem.setName("ViewGraphMenuItem");
						ivjViewGraphMenuItem.setText("View Graph in New Window");
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjViewGraphMenuItem;
}

/**
 * Return the JMenu2 property value.
 * @return  JMenu
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenu getViewMenu() {
		if (ivjViewMenu == null) {
				try {
						ivjViewMenu = new  JMenu();
						ivjViewMenu.setName("ViewMenu");
						ivjViewMenu.setText("View");
						ivjViewMenu.add(getViewGraphMenuItem());
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjViewMenu;
}

/**
 * Return the JMenuItem4 property value.
 * @return  JMenuItem
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JMenuItem getViewUserMenuItem() {
		if (ivjViewUserMenuItem == null) {
				try {
						ivjViewUserMenuItem = new  JMenuItem();
						ivjViewUserMenuItem.setName("ViewUserMenuItem");
						ivjViewUserMenuItem.setText("View User");
						// user code begin {1}
						// user code end
				} catch ( Throwable ivjExc) {
						// user code begin {2}
						// user code end
						handleException(ivjExc);
				}
		}
		return ivjViewUserMenuItem;
}
```

```
/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("MainFrame");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setJMenuBar(getMainFrameJMenuBar());
                setSize(852, 580);
                setTitle("Resource Management");
                setContentPane(getJFrameContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                MainFrame aMainFrame;
                aMainFrame = new MainFrame();
                aMainFrame.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aMainFrame.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JFrame");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 * 1/29/00 -  Changed the tree view so it was a tabbed pane in order to have the
 *                                           two separate trees - project and resource
 * 1/29/00 -   Added the start and end date fields above the graph
 *
 */
```

ModifyButtonsPanel.java

```
//------------------------------------------------------------------------------
// ModifyButtonsPanel.java
//
// Author: Mark Briggs
//                             John Schluechtermann
```

```
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/20/00
// Last Modified: 1/20/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//----------------------------------------------------------------------------

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

/**
 * A panel class that is used to group button together for display.
 * It holds the buttons for accepting and cancelling a change/addition made.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/
class ModifyButtonsPanel extends  JPanel {
       private  JButton ivjAcceptButton = null;
       private  JButton ivjCancelButton = null;

/**
 * ModifyButtonsPanel constructor comment.
 */
public ModifyButtonsPanel() {
       super();
       initialize();
}

/**
 * ModifyButtonsPanel constructor comment.
 * @param layout  LayoutManager
 */
public ModifyButtonsPanel( LayoutManager layout) {
       super(layout);
}

/**
 * ModifyButtonsPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public ModifyButtonsPanel( LayoutManager layout, boolean isDoubleBuffered) {
       super(layout, isDoubleBuffered);
}

/**
 * ModifyButtonsPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public ModifyButtonsPanel(boolean isDoubleBuffered) {
       super(isDoubleBuffered);
}

/**
 * Return the JButton1 property value.
 * @return  JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JButton getAcceptButton() {
       if (ivjAcceptButton == null) {
              try {
                     ivjAcceptButton = new  JButton();
                     ivjAcceptButton.setName("AcceptButton");
```

```
                        ivjAcceptButton.setText("Accept");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjAcceptButton;
}

/**
 * Return the JButton2 property value.
 * @return  JButton
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JButton getCancelButton() {
        if (ivjCancelButton == null) {
                try {
                        ivjCancelButton = new  JButton();
                        ivjCancelButton.setName("CancelButton");
                        ivjCancelButton.setText("Cancel");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjCancelButton;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.  This arranges the two buttons in the proper location
 * on the panel using gridbag layout.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("ModifyButtonsPanel");
                setLayout(new  GridBagLayout());
                setSize(242, 56);

                GridBagConstraints constraintsAcceptButton = new  GridBagConstraints();
                constraintsAcceptButton.gridx = 1; constraintsAcceptButton.gridy = 1;
                constraintsAcceptButton.ipadx = 10;
                constraintsAcceptButton.insets = new  Insets(15, 24, 16, 15);
                add(getAcceptButton(), constraintsAcceptButton);

                GridBagConstraints constraintsCancelButton = new  GridBagConstraints();
                constraintsCancelButton.gridx = 2; constraintsCancelButton.gridy = 1;
                constraintsCancelButton.ipadx = 12;
                constraintsCancelButton.insets = new  Insets(15, 16, 16, 17);
                add(getCancelButton(), constraintsCancelButton);
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                 JFrame frame = new  JFrame();
                ModifyButtonsPanel aModifyButtonsPanel;
                aModifyButtonsPanel = new ModifyButtonsPanel();
                frame.setContentPane(aModifyButtonsPanel);
                frame.setSize(aModifyButtonsPanel.getSize());
                frame.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                frame.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JPanel");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

ProjectFormPanel.java

```
//-------------------------------------------------------------------------------
// ProjectFormPanel.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/19/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//-------------------------------------------------------------------------------

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

/**
 * A panel class that is used with the information regarding a project
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modification of the project.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class ProjectFormPanel extends  JPanel {
// Private data members
```

```
        private  JTextField ivjProjectNameTextField = null;
        private ModifyButtonsPanel ivjModifyButtonsPanel1 = null;
        private  JLabel ivjClientLabel = null;
        private  JLabel ivjProjectNameLabel = null;
        private  JLabel ivjStatusLabel = null;
        private  JLabel ivjHeadingLabel = null;
        private  JComboBox ivjClientComboBox = null;
        private  JComboBox ivjStatusComboBox = null;

/**
 * ProjectFormPanel constructor comment.
 */
public ProjectFormPanel() {
        super();
        initialize();
}

/**
 * ProjectFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public ProjectFormPanel( LayoutManager layout) {
        super(layout);
}

/**
 * ProjectFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public ProjectFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
        super(layout, isDoubleBuffered);
}

/**
 * ProjectFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public ProjectFormPanel(boolean isDoubleBuffered) {
        super(isDoubleBuffered);
}

/**
 * Return the JComboBox1 property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getClientComboBox() {
        if (ivjClientComboBox == null) {
                try {
                        ivjClientComboBox = new  JComboBox();
                        ivjClientComboBox.setName("ClientComboBox");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjClientComboBox;
}

/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getClientLabel() {
        if (ivjClientLabel == null) {
                try {
                        ivjClientLabel = new  JLabel();
                        ivjClientLabel.setName("ClientLabel");
                        ivjClientLabel.setText("Client");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjClientLabel;
}

/**
 * Return the JLabel4 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
        if (ivjHeadingLabel == null) {
                try {
                        ivjHeadingLabel = new  JLabel();
                        ivjHeadingLabel.setName("HeadingLabel");
                        ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
                        ivjHeadingLabel.setText("Project Information");
                        ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHeadingLabel;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel1() {
        if (ivjModifyButtonsPanel1 == null) {
                try {
                        ivjModifyButtonsPanel1 = new ModifyButtonsPanel();
                        ivjModifyButtonsPanel1.setName("ModifyButtonsPanel1");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjModifyButtonsPanel1;
}

/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getProjectNameLabel() {
        if (ivjProjectNameLabel == null) {
                try {
                        ivjProjectNameLabel = new  JLabel();
                        ivjProjectNameLabel.setName("ProjectNameLabel");
                        ivjProjectNameLabel.setText("Project Name");
                        // user code begin {1}
```

```
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjProjectNameLabel;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getProjectNameTextField() {
		if (ivjProjectNameTextField == null) {
			try {
				ivjProjectNameTextField = new  JTextField();
				ivjProjectNameTextField.setName("ProjectNameTextField");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjProjectNameTextField;
}

/**
 * Return the JComboBox2 property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getStatusComboBox() {
		if (ivjStatusComboBox == null) {
			try {
				ivjStatusComboBox = new  JComboBox();
				ivjStatusComboBox.setName("StatusComboBox");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
		return ivjStatusComboBox;
}

/**
 * Return the JLabel3 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getStatusLabel() {
		if (ivjStatusLabel == null) {
			try {
				ivjStatusLabel = new  JLabel();
				ivjStatusLabel.setName("StatusLabel");
				ivjStatusLabel.setText("Status");
				// user code begin {1}
				// user code end
			} catch ( Throwable ivjExc) {
				// user code begin {2}
				// user code end
				handleException(ivjExc);
			}
		}
```

```
        return ivjStatusLabel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.  This position all the peices that make up the
 * panel using gridbag layout.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("ProjectFormPanel");
                setLayout(new  GridBagLayout());
                setSize(442, 344);

                GridBagConstraints constraintsProjectNameLabel = new  GridBagConstraints();
                constraintsProjectNameLabel.gridx = 1; constraintsProjectNameLabel.gridy = 2;
                constraintsProjectNameLabel.gridwidth = 2;
                constraintsProjectNameLabel.ipadx = 14;
                constraintsProjectNameLabel.insets = new  Insets(19, 52, 13, 8);
                add(getProjectNameLabel(), constraintsProjectNameLabel);

                GridBagConstraints constraintsClientLabel = new  GridBagConstraints();
                constraintsClientLabel.gridx = 1; constraintsClientLabel.gridy = 3;
                constraintsClientLabel.ipadx = 13;
                constraintsClientLabel.insets = new  Insets(18, 52, 16, 1);
                add(getClientLabel(), constraintsClientLabel);

                GridBagConstraints constraintsStatusLabel = new  GridBagConstraints();
                constraintsStatusLabel.gridx = 1; constraintsStatusLabel.gridy = 4;
                constraintsStatusLabel.ipadx = 8;
                constraintsStatusLabel.insets = new  Insets(17, 52, 20, 1);
                add(getStatusLabel(), constraintsStatusLabel);

                GridBagConstraints constraintsProjectNameTextField = new  GridBagConstraints();
                constraintsProjectNameTextField.gridx = 3; constraintsProjectNameTextField.gridy =
                2;
                constraintsProjectNameTextField.fill =  GridBagConstraints.HORIZONTAL;
                constraintsProjectNameTextField.weightx = 1.0;
                constraintsProjectNameTextField.ipadx = 230;
                constraintsProjectNameTextField.insets = new  Insets(17, 9, 11, 48);
                add(getProjectNameTextField(), constraintsProjectNameTextField);

                GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
                constraintsHeadingLabel.gridx = 1; constraintsHeadingLabel.gridy = 1;
                constraintsHeadingLabel.gridwidth = 3;
                constraintsHeadingLabel.ipadx = 90;
                constraintsHeadingLabel.ipady = 28;
                constraintsHeadingLabel.insets = new  Insets(31, 65, 16, 66);
                add(getHeadingLabel(), constraintsHeadingLabel);

                GridBagConstraints constraintsModifyButtonsPanel1 = new  GridBagConstraints();
                constraintsModifyButtonsPanel1.gridx = 2; constraintsModifyButtonsPanel1.gridy =
                5;
                constraintsModifyButtonsPanel1.gridwidth = 2;
                constraintsModifyButtonsPanel1.fill =  GridBagConstraints.BOTH;
                constraintsModifyButtonsPanel1.weightx = 1.0;
                constraintsModifyButtonsPanel1.weighty = 1.0;
                constraintsModifyButtonsPanel1.insets = new  Insets(14, 2, 22, 100);
```

```
		add(getModifyButtonsPanel1(), constraintsModifyButtonsPanel1);

		GridBagConstraints constraintsClientComboBox = new  GridBagConstraints();
		constraintsClientComboBox.gridx = 3; constraintsClientComboBox.gridy = 3;
		constraintsClientComboBox.fill =  GridBagConstraints.HORIZONTAL;
		constraintsClientComboBox.weightx = 1.0;
		constraintsClientComboBox.ipadx = 4;
		constraintsClientComboBox.insets = new  Insets(12, 9, 10, 152);
		add(getClientComboBox(), constraintsClientComboBox);

		GridBagConstraints constraintsStatusComboBox = new  GridBagConstraints();
		constraintsStatusComboBox.gridx = 3; constraintsStatusComboBox.gridy = 4;
		constraintsStatusComboBox.fill =  GridBagConstraints.HORIZONTAL;
		constraintsStatusComboBox.weightx = 1.0;
		constraintsStatusComboBox.ipadx = 4;
		constraintsStatusComboBox.insets = new  Insets(11, 9, 14, 152);
		add(getStatusComboBox(), constraintsStatusComboBox);
	} catch ( Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
	try {
		 JFrame frame = new  JFrame();
		ProjectFormPanel aProjectFormPanel;
		aProjectFormPanel = new ProjectFormPanel();
		frame.setContentPane(aProjectFormPanel);
		frame.setSize(aProjectFormPanel.getSize());
		frame.addWindowListener(new  WindowAdapter() {
			public void windowClosing( WindowEvent e) {
				System.exit(0);
			};
		});
		frame.setVisible(true);
	} catch (Throwable exception) {
		System.err.println("Exception occurred in main() of  JPanel");
		exception.printStackTrace(System.out);
	}
}
}

/*
 * Revision History
 *
 */
```

RoomFormPanel.java

```
//--------------------------------------------------------------------------
// RoomFormPanel.java
//
// Author: Mark Briggs
//                             John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/19/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
```

```
//---------------------------------------------------------------------------

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

/**
 * A panel class that is used with the information for a room.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for viewing and modifying the room information.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class RoomFormPanel extends  JPanel {
       private DefaultRatePanel ivjDefaultRatePanel = null;
       private  JLabel ivjDefaultsLabel = null;
       private  JLabel ivjDescriptionLabel = null;
       private  JTextArea ivjDescriptionTextField = null;
       private  JLabel ivjHeadingLabel = null;
       private  JPanel ivjInfoPanel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private  JLabel ivjNameLabel = null;
       private  JTextField ivjNameTextField = null;

/**
 * RoomFormPanel constructor comment.
 */
public RoomFormPanel() {
       super();
       initialize();
}

/**
 * RoomFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public RoomFormPanel( LayoutManager layout) {
       super(layout);
}

/**
 * RoomFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public RoomFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
       super(layout, isDoubleBuffered);
}

/**
 * RoomFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public RoomFormPanel(boolean isDoubleBuffered) {
       super(isDoubleBuffered);
}

/**
 * Return the DefaultRatePanel1 property value.
 * @return DefaultRatePanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private DefaultRatePanel getDefaultRatePanel() {
       if (ivjDefaultRatePanel == null) {
              try {
                     ivjDefaultRatePanel = new DefaultRatePanel();
                     ivjDefaultRatePanel.setName("DefaultRatePanel");
```

```
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjDefaultRatePanel;
}

/**
 * Return the JLabel1 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getDefaultsLabel() {
        if (ivjDefaultsLabel == null) {
                try {
                        ivjDefaultsLabel = new  JLabel();
                        ivjDefaultsLabel.setName("DefaultsLabel");
                        ivjDefaultsLabel.setText("Defaults");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjDefaultsLabel;
}

/**
 * Return the JLabel4 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getDescriptionLabel() {
        if (ivjDescriptionLabel == null) {
                try {
                        ivjDescriptionLabel = new  JLabel();
                        ivjDescriptionLabel.setName("DescriptionLabel");
                        ivjDescriptionLabel.setText("Description");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjDescriptionLabel;
}

/**
 * Return the JTextArea1 property value.
 * @return  JTextArea
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextArea getDescriptionTextField() {
        if (ivjDescriptionTextField == null) {
                try {
                        ivjDescriptionTextField = new  JTextArea();
                        ivjDescriptionTextField.setName("DescriptionTextField");
                        ivjDescriptionTextField.setLineWrap(true);
                        ivjDescriptionTextField.setRows(4);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjDescriptionTextField;
}

/**
 * Return the JLabel2 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
        if (ivjHeadingLabel == null) {
                try {
                        ivjHeadingLabel = new  JLabel();
                        ivjHeadingLabel.setName("HeadingLabel");
                        ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
                        ivjHeadingLabel.setText("Room Information");
                        ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHeadingLabel;
}

/**
 * Return the JPanel1 property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getInfoPanel() {
        if (ivjInfoPanel == null) {
                try {
                        ivjInfoPanel = new  JPanel();
                        ivjInfoPanel.setName("InfoPanel");
                        ivjInfoPanel.setLayout(new  GridBagLayout());

                        GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
                        constraintsNameLabel.gridx = 0; constraintsNameLabel.gridy = 0;
                        constraintsNameLabel.ipadx = 12;
                        constraintsNameLabel.insets = new  Insets(28, 28, 6, 51);
                        getInfoPanel().add(getNameLabel(), constraintsNameLabel);

                        GridBagConstraints constraintsDescriptionLabel = new
                        GridBagConstraints();
                        constraintsDescriptionLabel.gridx = 0; constraintsDescriptionLabel.gridy =
                        1;
                        constraintsDescriptionLabel.ipadx = 20;
                        constraintsDescriptionLabel.insets = new  Insets(29, 28, 37, 11);
                        getInfoPanel().add(getDescriptionLabel(), constraintsDescriptionLabel);

                        GridBagConstraints constraintsDescriptionTextField = new
                        GridBagConstraints();
                        constraintsDescriptionTextField.gridx = 1;
                        constraintsDescriptionTextField.gridy = 1;
                        constraintsDescriptionTextField.fill =  GridBagConstraints.BOTH;
                        constraintsDescriptionTextField.weightx = 1.0;
                        constraintsDescriptionTextField.weighty = 1.0;
                        constraintsDescriptionTextField.ipadx = 194;
                        constraintsDescriptionTextField.ipady = 40;
                        constraintsDescriptionTextField.insets = new  Insets(5, 12, 17, 43);
                        getInfoPanel().add(getDescriptionTextField(),
                        constraintsDescriptionTextField);

                        GridBagConstraints constraintsNameTextField = new  GridBagConstraints();
```

```
			constraintsNameTextField.gridx = 1; constraintsNameTextField.gridy = 0;
			constraintsNameTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsNameTextField.weightx = 1.0;
			constraintsNameTextField.ipadx = 205;
			constraintsNameTextField.insets = new  Insets(26, 12, 4, 43);
			getInfoPanel().add(getNameTextField(), constraintsNameTextField);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjInfoPanel;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
	if (ivjModifyButtonsPanel == null) {
		try {
			ivjModifyButtonsPanel = new ModifyButtonsPanel();
			ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjModifyButtonsPanel;
}

/**
 * Return the JLabel3 property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
	if (ivjNameLabel == null) {
		try {
			ivjNameLabel = new  JLabel();
			ivjNameLabel.setName("NameLabel");
			ivjNameLabel.setText("Name");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameLabel;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
	if (ivjNameTextField == null) {
		try {
			ivjNameTextField = new  JTextField();
			ivjNameTextField.setName("NameTextField");
			// user code begin {1}
```

```
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNameTextField;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("RoomFormPanel");
                setLayout(new  GridBagLayout());
                setSize(456, 558);

                GridBagConstraints constraintsDefaultRatePanel = new  GridBagConstraints();
                constraintsDefaultRatePanel.gridx = 1; constraintsDefaultRatePanel.gridy = 4;
                constraintsDefaultRatePanel.gridwidth = 2;
                constraintsDefaultRatePanel.fill =  GridBagConstraints.BOTH;
                constraintsDefaultRatePanel.weightx = 1.0;
                constraintsDefaultRatePanel.weighty = 1.0;
                constraintsDefaultRatePanel.ipadx = -4;
                constraintsDefaultRatePanel.ipady = -11;
                constraintsDefaultRatePanel.insets = new  Insets(1, 30, 5, 38);
                add(getDefaultRatePanel(), constraintsDefaultRatePanel);

                GridBagConstraints constraintsDefaultsLabel = new  GridBagConstraints();
                constraintsDefaultsLabel.gridx = 1; constraintsDefaultsLabel.gridy = 3;
                constraintsDefaultsLabel.ipadx = 10;
                constraintsDefaultsLabel.insets = new  Insets(4, 30, 0, 10);
                add(getDefaultsLabel(), constraintsDefaultsLabel);

                GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
                constraintsHeadingLabel.gridx = 2; constraintsHeadingLabel.gridy = 1;
                constraintsHeadingLabel.ipadx = 27;
                constraintsHeadingLabel.ipady = 12;
                constraintsHeadingLabel.insets = new  Insets(23, 13, 11, 111);
                add(getHeadingLabel(), constraintsHeadingLabel);

                GridBagConstraints constraintsInfoPanel = new  GridBagConstraints();
                constraintsInfoPanel.gridx = 1; constraintsInfoPanel.gridy = 2;
                constraintsInfoPanel.gridwidth = 2;
                constraintsInfoPanel.fill =  GridBagConstraints.BOTH;
                constraintsInfoPanel.weightx = 1.0;
                constraintsInfoPanel.weighty = 1.0;
                constraintsInfoPanel.ipadx = -194;
                constraintsInfoPanel.ipady = -7;
                constraintsInfoPanel.insets = new  Insets(12, 30, 3, 38);
                add(getInfoPanel(), constraintsInfoPanel);

                GridBagConstraints constraintsModifyButtonsPanel = new  GridBagConstraints();
                constraintsModifyButtonsPanel.gridx = 2; constraintsModifyButtonsPanel.gridy = 5;
                constraintsModifyButtonsPanel.fill =  GridBagConstraints.BOTH;
```

```
			constraintsModifyButtonsPanel.weightx = 1.0;
			constraintsModifyButtonsPanel.weighty = 1.0;
			constraintsModifyButtonsPanel.insets = new  Insets(6, 10, 16, 107);
			add(getModifyButtonsPanel(), constraintsModifyButtonsPanel);
		} catch ( Throwable ivjExc) {
			handleException(ivjExc);
		}
		// user code begin {2}
		// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
		try {
			 JFrame frame = new  JFrame();
			RoomFormPanel aRoomFormPanel;
			aRoomFormPanel = new RoomFormPanel();
			frame.setContentPane(aRoomFormPanel);
			frame.setSize(aRoomFormPanel.getSize());
			frame.addWindowListener(new  WindowAdapter() {
				public void windowClosing( WindowEvent e) {
					System.exit(0);
				};
			});
			frame.setVisible(true);
		} catch (Throwable exception) {
			System.err.println("Exception occurred in main() of  JPanel");
			exception.printStackTrace(System.out);
		}
}
}

/*
 * Revision History
 *
 */
```

TimeslotFormPanel.java

```
//------------------------------------------------------------------------------
// TimeslotFormPanel.java
//
// Author: Mark Briggs
//                            John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/19/00
// Last Modified: 1/29/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//------------------------------------------------------------------------------


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

/**
 * A panel class that is used to arrange the timeslot information
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modification of the timeslot.
 *
 * @version   1.1
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class TimeslotFormPanel extends JPanel {
       // Private data members
       private JLabel ivjEndDateLabel = null;
       private JTextField ivjEndDateTextField = null;
       private JLabel ivjHeadingLabel = null;
       private JLabel ivjHoursDayLabel = null;
       private JTextField ivjHoursDayTextField = null;
       private JPanel ivjInfoPanel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private JLabel ivjOfficeLabel = null;
       private JLabel ivjOvertimeRateLabe = null;
       private JTextField ivjOvertimeRateTextField = null;
       private JTextField ivjRegularRateTextField = null;
       private JLabel ivjStartDateLabel = null;
       private JTextField ivjStartDateTextField = null;
       private JComboBox ivjOfficeComboBox = null;
       private JLabel ivjallocRateLabel = null;
       private JLabel ivjcostLabel = null;
       private JTextField ivjcostTextField = null;
       private JLabel ivjovertimeCostRateLabel = null;
       private JTextField ivjovertimeCostTextField = null;

/**
 * AllocationFormPanel constructor comment.
 */
public TimeslotFormPanel() {
       super();
       initialize();
}

/**
 * AllocationFormPanel constructor comment.
 * @param layout LayoutManager
 */
public TimeslotFormPanel(LayoutManager layout) {
       super(layout);
}

/**
 * AllocationFormPanel constructor comment.
 * @param layout LayoutManager
 * @param isDoubleBuffered boolean
 */
public TimeslotFormPanel(LayoutManager layout, boolean isDoubleBuffered) {
       super(layout, isDoubleBuffered);
}

/**
 * AllocationFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public TimeslotFormPanel(boolean isDoubleBuffered) {
       super(isDoubleBuffered);
}

/**
 * Return the JLabel4 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getallocRateLabel() {
       if (ivjallocRateLabel == null) {
              try {
                     ivjallocRateLabel = new JLabel();
                     ivjallocRateLabel.setName("allocRateLabel");
                     ivjallocRateLabel.setText("Allocated Rate");
```

```
			ivjallocRateLabel.setBounds(43, 117, 91, 15);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjallocRateLabel;
}

/**
 * Return the costLabel property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getcostLabel() {
	if (ivjcostLabel == null) {
		try {
			ivjcostLabel = new JLabel();
			ivjcostLabel.setName("costLabel");
			ivjcostLabel.setText("Minimum Rate");
			ivjcostLabel.setBounds(43, 150, 89, 15);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjcostLabel;
}

/**
 * Return the costTextField property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getcostTextField() {
	if (ivjcostTextField == null) {
		try {
			ivjcostTextField = new JTextField();
			ivjcostTextField.setName("costTextField");
			ivjcostTextField.setBounds(198, 146, 190, 19);
			ivjcostTextField.setEditable(false);
			ivjcostTextField.setRequestFocusEnabled(false);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjcostTextField;
}

/**
 * Return the JLabel21 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getEndDateLabel() {
	if (ivjEndDateLabel == null) {
		try {
			ivjEndDateLabel = new JLabel();
			ivjEndDateLabel.setName("EndDateLabel");
			ivjEndDateLabel.setText("End Date");
			ivjEndDateLabel.setBounds(43, 51, 65, 15);
```

```
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjEndDateLabel;
}

/**
 * Return the JTextField2 property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getEndDateTextField() {
        if (ivjEndDateTextField == null) {
                try {
                        ivjEndDateTextField = new JTextField();
                        ivjEndDateTextField.setName("EndDateTextField");
                        ivjEndDateTextField.setBounds(198, 47, 190, 19);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjEndDateTextField;
}

/**
 * Return the JLabel1 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getHeadingLabel() {
        if (ivjHeadingLabel == null) {
                try {
                        ivjHeadingLabel = new JLabel();
                        ivjHeadingLabel.setName("HeadingLabel");
                        ivjHeadingLabel.setFont(new Font("dialog", 1, 24));
                        ivjHeadingLabel.setText("Timeslot Information");
                        ivjHeadingLabel.setBounds(170, 12, 282, 47);
                        ivjHeadingLabel.setHorizontalAlignment(SwingConstants.CENTER);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHeadingLabel;
}

/**
 * Return the JLabel3 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getHoursDayLabel() {
        if (ivjHoursDayLabel == null) {
                try {
                        ivjHoursDayLabel = new JLabel();
                        ivjHoursDayLabel.setName("HoursDayLabel");
                        ivjHoursDayLabel.setText("Hours / Day");
                        ivjHoursDayLabel.setBounds(43, 84, 75, 15);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHoursDayLabel;
}

/**
 * Return the JTextField3 property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getHoursDayTextField() {
        if (ivjHoursDayTextField == null) {
                try {
                        ivjHoursDayTextField = new JTextField();
                        ivjHoursDayTextField.setName("HoursDayTextField");
                        ivjHoursDayTextField.setBounds(198, 80, 190, 19);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjHoursDayTextField;
}

/**
 * Return the JPanel1 property value.
 * @return JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JPanel getInfoPanel() {
        if (ivjInfoPanel == null) {
                try {
                        ivjInfoPanel = new JPanel();
                        ivjInfoPanel.setName("InfoPanel");
                        ivjInfoPanel.setLayout(null);
                        ivjInfoPanel.setBounds(23, 66, 511, 288);
                        getInfoPanel().add(getStartDateLabel(), getStartDateLabel().getName());
                        getInfoPanel().add(getStartDateTextField(),
                        getStartDateTextField().getName());
                        getInfoPanel().add(getEndDateLabel(), getEndDateLabel().getName());
                        getInfoPanel().add(getEndDateTextField(),
                        getEndDateTextField().getName());
                        getInfoPanel().add(getHoursDayLabel(), getHoursDayLabel().getName());
                        getInfoPanel().add(getHoursDayTextField(),
                        getHoursDayTextField().getName());
                        getInfoPanel().add(getallocRateLabel(), getallocRateLabel().getName());
                        getInfoPanel().add(getRegularRateTextField(),
                        getRegularRateTextField().getName());
                        getInfoPanel().add(getOvertimeRateLabe(),
                        getOvertimeRateLabe().getName());
                        getInfoPanel().add(getOvertimeRateTextField(),
                        getOvertimeRateTextField().getName());
                        getInfoPanel().add(getOfficeLabel(), getOfficeLabel().getName());
                        getInfoPanel().add(getOfficeComboBox(), getOfficeComboBox().getName());
                        getInfoPanel().add(getcostLabel(), getcostLabel().getName());
                        getInfoPanel().add(getovertimeCostRateLabel(),
                        getovertimeCostRateLabel().getName());
                        getInfoPanel().add(getcostTextField(), getcostTextField().getName());
                        getInfoPanel().add(getovertimeCostTextField(),
                        getovertimeCostTextField().getName());
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjInfoPanel;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
        if (ivjModifyButtonsPanel == null) {
                try {
                        ivjModifyButtonsPanel = new ModifyButtonsPanel();
                        ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
                        ivjModifyButtonsPanel.setBounds(165, 356, 242, 73);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjModifyButtonsPanel;
}

/**
 * Return the JComboBox1 property value.
 * @return JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JComboBox getOfficeComboBox() {
        if (ivjOfficeComboBox == null) {
                try {
                        ivjOfficeComboBox = new JComboBox();
                        ivjOfficeComboBox.setName("OfficeComboBox");
                        ivjOfficeComboBox.setBounds(198, 245, 130, 27);
                        ivjOfficeComboBox.setRequestFocusEnabled(false);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjOfficeComboBox;
}

/**
 * Return the JLabel8 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getOfficeLabel() {
        if (ivjOfficeLabel == null) {
                try {
                        ivjOfficeLabel = new JLabel();
                        ivjOfficeLabel.setName("OfficeLabel");
                        ivjOfficeLabel.setText("Office");
                        ivjOfficeLabel.setBounds(43, 249, 45, 15);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjOfficeLabel;
}

/**
 * Return the overtimeCostRateLabel property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getovertimeCostRateLabel() {
        if (ivjovertimeCostRateLabel == null) {
                try {
                        ivjovertimeCostRateLabel = new JLabel();
                        ivjovertimeCostRateLabel.setName("overtimeCostRateLabel");
                        ivjovertimeCostRateLabel.setText("Minimum Overtime Rate");
                        ivjovertimeCostRateLabel.setBounds(43, 216, 139, 15);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjovertimeCostRateLabel;
}

/**
 * Return the overtimeCostTextField property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getovertimeCostTextField() {
        if (ivjovertimeCostTextField == null) {
                try {
                        ivjovertimeCostTextField = new JTextField();
                        ivjovertimeCostTextField.setName("overtimeCostTextField");
                        ivjovertimeCostTextField.setBounds(198, 212, 190, 19);
                        ivjovertimeCostTextField.setEditable(false);
                        ivjovertimeCostTextField.setRequestFocusEnabled(false);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjovertimeCostTextField;
}

/**
 * Return the JLabel5 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getOvertimeRateLabe() {
        if (ivjOvertimeRateLabe == null) {
                try {
                        ivjOvertimeRateLabe = new JLabel();
                        ivjOvertimeRateLabe.setName("OvertimeRateLabe");
                        ivjOvertimeRateLabe.setText("Allocated Overtime Rate");
                        ivjOvertimeRateLabe.setBounds(43, 183, 139, 15);
                        // user code begin {1}
                        // user code end
                } catch (Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
```

```
		}
		return ivjOvertimeRateLabe;
}

/**
 * Return the JTextField5 property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getOvertimeRateTextField() {
	if (ivjOvertimeRateTextField == null) {
		try {
			ivjOvertimeRateTextField = new JTextField();
			ivjOvertimeRateTextField.setName("OvertimeRateTextField");
			ivjOvertimeRateTextField.setBounds(198, 179, 190, 19);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjOvertimeRateTextField;
}

/**
 * Return the JTextField4 property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getRegularRateTextField() {
	if (ivjRegularRateTextField == null) {
		try {
			ivjRegularRateTextField = new JTextField();
			ivjRegularRateTextField.setName("RegularRateTextField");
			ivjRegularRateTextField.setBounds(198, 113, 190, 19);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjRegularRateTextField;
}

/**
 * Return the JLabel2 property value.
 * @return JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JLabel getStartDateLabel() {
	if (ivjStartDateLabel == null) {
		try {
			ivjStartDateLabel = new JLabel();
			ivjStartDateLabel.setName("StartDateLabel");
			ivjStartDateLabel.setText("Start Date");
			ivjStartDateLabel.setBounds(43, 18, 75, 15);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjStartDateLabel;
}
```

```
/**
 * Return the JTextField1 property value.
 * @return JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private JTextField getStartDateTextField() {
	if (ivjStartDateTextField == null) {
		try {
			ivjStartDateTextField = new JTextField();
			ivjStartDateTextField.setName("StartDateTextField");
			ivjStartDateTextField.setBounds(198, 14, 190, 19);
			// user code begin {1}
			// user code end
		} catch (Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjStartDateTextField;
}

/**
 * Called whenever the part throws an exception.
 * @param exception Throwable
 */
private void handleException(Throwable exception) {

	/* Uncomment the following lines to print uncaught exceptions to stdout */
	// System.out.println("--------- UNCAUGHT EXCEPTION ---------");
	// exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
	try {
		// user code begin {1}
		// user code end
		setName("AllocationFormPanel");
		setLayout(null);
		setSize(572, 440);
		add(getHeadingLabel(), getHeadingLabel().getName());
		add(getInfoPanel(), getInfoPanel().getName());
		add(getModifyButtonsPanel(), getModifyButtonsPanel().getName());
	} catch (Throwable ivjExc) {
		handleException(ivjExc);
	}
	// user code begin {2}
	// user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args String[]
 */
public static void main(String[] args) {
	try {
		JFrame frame = new JFrame();
		TimeslotFormPanel aTimeslotFormPanel;
		aTimeslotFormPanel = new TimeslotFormPanel();
		frame.setContentPane(aTimeslotFormPanel);
		frame.setSize(aTimeslotFormPanel.getSize());
		frame.addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e) {
				System.exit(0);
			};
		});
		frame.setVisible(true);
```

```
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of JPanel");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 * 1/29/00 -   Eliminated the selection of the project and client and what type
 *
 */
```

UserDialog.java

```
//-----------------------------------------------------------------------------
// UserDialog.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 2/4/00
// Last Modified: 2/4/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//-----------------------------------------------------------------------------

import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * A dialog class that is used with viewing and modifying the user information.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modification of the user.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class UserDialog extends  JDialog {
        private  JPanel ivjJDialogContentPane = null;
        private  JComboBox ivjNameComboBox = null;
        private  JLabel ivjNameLabel = null;
        private  JPanel ivjNamePanel = null;
        private UserFormPanel ivjUserFormPanel = null;

/**
 * UserDialog constructor comment.
 */
public UserDialog() {
        super();
        initialize();
}

/**
 * UserDialog constructor comment.
 * @param owner  Frame
 */
public UserDialog( Frame owner) {
        super(owner);
}
```

```
/**
 * UserDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 */
public UserDialog( Frame owner, String title) {
	super(owner, title);
}

/**
 * UserDialog constructor comment.
 * @param owner  Frame
 * @param title  String
 * @param modal boolean
 */
public UserDialog( Frame owner, String title, boolean modal) {
	super(owner, title, modal);
}

/**
 * UserDialog constructor comment.
 * @param owner  Frame
 * @param modal boolean
 */
public UserDialog( Frame owner, boolean modal) {
	super(owner, modal);
}

/**
 * Return the JDialogContentPane property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getJDialogContentPane() {
	if (ivjJDialogContentPane == null) {
		try {
			ivjJDialogContentPane = new  JPanel();
			ivjJDialogContentPane.setName("JDialogContentPane");
			ivjJDialogContentPane.setLayout(null);
			getJDialogContentPane().add(getUserFormPanel(),
			getUserFormPanel().getName());
			getJDialogContentPane().add(getNamePanel(), getNamePanel().getName());
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjJDialogContentPane;
}

/**
 * Return the NameComboBox property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getNameComboBox() {
	if (ivjNameComboBox == null) {
		try {
			ivjNameComboBox = new  JComboBox();
			ivjNameComboBox.setName("NameComboBox");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
```

```
        }
        return ivjNameComboBox;
}

/**
 * Return the NameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
        if (ivjNameLabel == null) {
                try {
                        ivjNameLabel = new  JLabel();
                        ivjNameLabel.setName("NameLabel");
                        ivjNameLabel.setText("Name");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNameLabel;
}

/**
 * Return the NamePanel property value.
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getNamePanel() {
        if (ivjNamePanel == null) {
                try {
                        ivjNamePanel = new  JPanel();
                        ivjNamePanel.setName("NamePanel");
                        ivjNamePanel.setLayout(new  GridBagLayout());
                        ivjNamePanel.setBounds(19, 20, 262, 64);

                        GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
                        constraintsNameLabel.gridx = 1; constraintsNameLabel.gridy = 1;
                        constraintsNameLabel.ipadx = 12;
                        constraintsNameLabel.insets = new  Insets(24, 22, 25, 20);
                        getNamePanel().add(getNameLabel(), constraintsNameLabel);

                        GridBagConstraints constraintsNameComboBox = new  GridBagConstraints();
                        constraintsNameComboBox.gridx = 2; constraintsNameComboBox.gridy = 1;
                        constraintsNameComboBox.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsNameComboBox.weightx = 1.0;
                        constraintsNameComboBox.ipadx = 4;
                        constraintsNameComboBox.insets = new  Insets(18, 21, 19, 20);
                        getNamePanel().add(getNameComboBox(), constraintsNameComboBox);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNamePanel;
}

/**
 * Return the UserFormPanel property value.
 * @return UserFormPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private UserFormPanel getUserFormPanel() {
        if (ivjUserFormPanel == null) {
                try {
                        ivjUserFormPanel = new UserFormPanel();
                        ivjUserFormPanel.setName("UserFormPanel");
                        ivjUserFormPanel.setBorder(new EtchedBorder());
                        ivjUserFormPanel.setLocation(19, 98);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjUserFormPanel;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("UserDialog");
                setDefaultCloseOperation( WindowConstants.DISPOSE_ON_CLOSE);
                setSize(488, 440);
                setTitle("User Information");
                setContentPane(getJDialogContentPane());
        } catch ( Throwable ivjExc) {
                handleException(ivjExc);
        }
        // user code begin {2}
        // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
        try {
                UserDialog aUserDialog;
                aUserDialog = new UserDialog();
                aUserDialog.setModal(true);
                aUserDialog.addWindowListener(new  WindowAdapter() {
                        public void windowClosing( WindowEvent e) {
                                System.exit(0);
                        };
                });
                aUserDialog.setVisible(true);
        } catch (Throwable exception) {
                System.err.println("Exception occurred in main() of  JDialog");
                exception.printStackTrace(System.out);
        }
}
}

/*
 * Revision History
 *
 */
```

UserFormPanel.java

```
//---------------------------------------------------------------------------
// UserFormPanel.java
//
// Author: Mark Briggs
//                              John Schluechtermann
//
// GUI created graphically and code generated from VisualAge Version 3.0
//
// Date created: 1/20/00
// Last Modified: 1/20/00
//
// Copyright (c) 2000 Resource Management Senior Design Group
// All rights reserved
//
//---------------------------------------------------------------------------

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


/**
 * A panel class that is used to arrange the user information.
 * It creates the necessary form that will be able to obtain all the necessary
 * information for the viewing and modification of the user.
 *
 * @version   1.0
 * @author            Mark Briggs
 * @author            John Schluechtermann
 **/

class UserFormPanel extends  JPanel {
       private  JLabel ivjAccessLevelLabel = null;
       private  JLabel ivjHeadingLabel = null;
       private  JPanel ivjInfoPanel = null;
       private ModifyButtonsPanel ivjModifyButtonsPanel = null;
       private  JLabel ivjNameLabel = null;
       private  JTextField ivjNameTextField = null;
       private  JLabel ivjPasswordLabel = null;
       private  JLabel ivjUserNameLabel = null;
       private  JTextField ivjUserNameTextField = null;
       private  JPasswordField ivjPasswordField = null;
       private  JComboBox ivjAccessLevelComboBox = null;

/**
 * UserFormPanel constructor comment.
 */
public UserFormPanel() {
       super();
       initialize();
}

/**
 * UserFormPanel constructor comment.
 * @param layout  LayoutManager
 */
public UserFormPanel( LayoutManager layout) {
       super(layout);
}

/**
 * UserFormPanel constructor comment.
 * @param layout  LayoutManager
 * @param isDoubleBuffered boolean
 */
public UserFormPanel( LayoutManager layout, boolean isDoubleBuffered) {
	super(layout, isDoubleBuffered);
}

/**
 * UserFormPanel constructor comment.
 * @param isDoubleBuffered boolean
 */
public UserFormPanel(boolean isDoubleBuffered) {
	super(isDoubleBuffered);
}

/**
 * Return the AccessLevelComboBox property value.
 * @return  JComboBox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JComboBox getAccessLevelComboBox() {
	if (ivjAccessLevelComboBox == null) {
		try {
			ivjAccessLevelComboBox = new  JComboBox();
			ivjAccessLevelComboBox.setName("AccessLevelComboBox");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAccessLevelComboBox;
}

/**
 * Return the AccessLevelLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getAccessLevelLabel() {
	if (ivjAccessLevelLabel == null) {
		try {
			ivjAccessLevelLabel = new  JLabel();
			ivjAccessLevelLabel.setName("AccessLevelLabel");
			ivjAccessLevelLabel.setText("Access Level");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjAccessLevelLabel;
}

/**
 * Return the HeadingLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getHeadingLabel() {
	if (ivjHeadingLabel == null) {
		try {
			ivjHeadingLabel = new  JLabel();
			ivjHeadingLabel.setName("HeadingLabel");
			ivjHeadingLabel.setFont(new  Font("dialog", 1, 24));
			ivjHeadingLabel.setText("User Information");
			ivjHeadingLabel.setHorizontalAlignment( SwingConstants.CENTER);
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjHeadingLabel;
}

/**
 * Return the InfoPanel property value.  Creates the panel and arranges the
 * layout components through gridbag layout.
 *
 * @return  JPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPanel getInfoPanel() {
	if (ivjInfoPanel == null) {
		try {
			ivjInfoPanel = new  JPanel();
			ivjInfoPanel.setName("InfoPanel");
			ivjInfoPanel.setLayout(new  GridBagLayout());

			GridBagConstraints constraintsNameLabel = new  GridBagConstraints();
			constraintsNameLabel.gridx = 1; constraintsNameLabel.gridy = 1;
			constraintsNameLabel.ipadx = 12;
			constraintsNameLabel.insets = new  Insets(21, 32, 10, 65);
			getInfoPanel().add(getNameLabel(), constraintsNameLabel);

			GridBagConstraints constraintsNameTextField = new  GridBagConstraints();
			constraintsNameTextField.gridx = 2; constraintsNameTextField.gridy = 1;
			constraintsNameTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsNameTextField.weightx = 1.0;
			constraintsNameTextField.ipadx = 164;
			constraintsNameTextField.insets = new  Insets(18, 16, 9, 46);
			getInfoPanel().add(getNameTextField(), constraintsNameTextField);

			GridBagConstraints constraintsUserNameLabel = new  GridBagConstraints();
			constraintsUserNameLabel.gridx = 1; constraintsUserNameLabel.gridy = 2;
			constraintsUserNameLabel.ipadx = 12;
			constraintsUserNameLabel.insets = new  Insets(11, 32, 10, 39);
			getInfoPanel().add(getUserNameLabel(), constraintsUserNameLabel);

			GridBagConstraints constraintsUserNameTextField = new
			GridBagConstraints();
			constraintsUserNameTextField.gridx = 2; constraintsUserNameTextField.gridy
			= 2;
			constraintsUserNameTextField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsUserNameTextField.weightx = 1.0;
			constraintsUserNameTextField.ipadx = 164;
			constraintsUserNameTextField.insets = new  Insets(9, 16, 8, 46);
			getInfoPanel().add(getUserNameTextField(), constraintsUserNameTextField);

			GridBagConstraints constraintsPasswordLabel = new  GridBagConstraints();
			constraintsPasswordLabel.gridx = 1; constraintsPasswordLabel.gridy = 3;
			constraintsPasswordLabel.ipadx = 31;
			constraintsPasswordLabel.insets = new  Insets(11, 32, 8, 21);
			getInfoPanel().add(getPasswordLabel(), constraintsPasswordLabel);

			GridBagConstraints constraintsAccessLevelLabel = new
			GridBagConstraints();
			constraintsAccessLevelLabel.gridx = 1; constraintsAccessLevelLabel.gridy =
			4;
			constraintsAccessLevelLabel.ipadx = 19;
			constraintsAccessLevelLabel.insets = new  Insets(13, 32, 22, 15);
			getInfoPanel().add(getAccessLevelLabel(), constraintsAccessLevelLabel);

			GridBagConstraints constraintsPasswordField = new  GridBagConstraints();
			constraintsPasswordField.gridx = 2; constraintsPasswordField.gridy = 3;
			constraintsPasswordField.fill =  GridBagConstraints.HORIZONTAL;
			constraintsPasswordField.weightx = 1.0;
```

```
                        constraintsPasswordField.ipadx = 164;
                        constraintsPasswordField.insets = new  Insets(9, 16, 6, 46);
                        getInfoPanel().add(getPasswordField(), constraintsPasswordField);

                        GridBagConstraints constraintsAccessLevelComboBox = new
                        GridBagConstraints();
                        constraintsAccessLevelComboBox.gridx = 2;
                        constraintsAccessLevelComboBox.gridy = 4;
                        constraintsAccessLevelComboBox.fill =  GridBagConstraints.HORIZONTAL;
                        constraintsAccessLevelComboBox.weightx = 1.0;
                        constraintsAccessLevelComboBox.ipadx = 4;
                        constraintsAccessLevelComboBox.insets = new  Insets(7, 16, 16, 84);
                        getInfoPanel().add(getAccessLevelComboBox(),
                        constraintsAccessLevelComboBox);
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjInfoPanel;
}

/**
 * Return the ModifyButtonsPanel1 property value.
 * @return ModifyButtonsPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private ModifyButtonsPanel getModifyButtonsPanel() {
        if (ivjModifyButtonsPanel == null) {
                try {
                        ivjModifyButtonsPanel = new ModifyButtonsPanel();
                        ivjModifyButtonsPanel.setName("ModifyButtonsPanel");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjModifyButtonsPanel;
}

/**
 * Return the NameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getNameLabel() {
        if (ivjNameLabel == null) {
                try {
                        ivjNameLabel = new  JLabel();
                        ivjNameLabel.setName("NameLabel");
                        ivjNameLabel.setText("Name");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjNameLabel;
}

/**
 * Return the JTextField1 property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getNameTextField() {
	if (ivjNameTextField == null) {
		try {
			ivjNameTextField = new  JTextField();
			ivjNameTextField.setName("NameTextField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjNameTextField;
}

/**
 * Return the PasswordField property value.
 * @return  JPasswordField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JPasswordField getPasswordField() {
	if (ivjPasswordField == null) {
		try {
			ivjPasswordField = new  JPasswordField();
			ivjPasswordField.setName("PasswordField");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjPasswordField;
}

/**
 * Return the PasswordLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getPasswordLabel() {
	if (ivjPasswordLabel == null) {
		try {
			ivjPasswordLabel = new  JLabel();
			ivjPasswordLabel.setName("PasswordLabel");
			ivjPasswordLabel.setText("Password");
			// user code begin {1}
			// user code end
		} catch ( Throwable ivjExc) {
			// user code begin {2}
			// user code end
			handleException(ivjExc);
		}
	}
	return ivjPasswordLabel;
}

/**
 * Return the UserNameLabel property value.
 * @return  JLabel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JLabel getUserNameLabel() {
	if (ivjUserNameLabel == null) {
		try {
			ivjUserNameLabel = new  JLabel();
			ivjUserNameLabel.setName("UserNameLabel");
```

```
                        ivjUserNameLabel.setText("Username");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjUserNameLabel;
}

/**
 * Return the UserNameTextField property value.
 * @return  JTextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private  JTextField getUserNameTextField() {
        if (ivjUserNameTextField == null) {
                try {
                        ivjUserNameTextField = new  JTextField();
                        ivjUserNameTextField.setName("UserNameTextField");
                        // user code begin {1}
                        // user code end
                } catch ( Throwable ivjExc) {
                        // user code begin {2}
                        // user code end
                        handleException(ivjExc);
                }
        }
        return ivjUserNameTextField;
}

/**
 * Called whenever the part throws an exception.
 * @param exception  Throwable
 */
private void handleException(Throwable exception) {

        /* Uncomment the following lines to print uncaught exceptions to stdout */
        // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
        // exception.printStackTrace(System.out);
}

/**
 * Initialize the class.  Used to position the elements that make up the
 * panel through the use of gridbag layout.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
        try {
                // user code begin {1}
                // user code end
                setName("UserFormPanel");
                setLayout(new  GridBagLayout());
                setSize(452, 332);

                GridBagConstraints constraintsInfoPanel = new  GridBagConstraints();
                constraintsInfoPanel.gridx = 1; constraintsInfoPanel.gridy = 2;
                constraintsInfoPanel.fill =  GridBagConstraints.BOTH;
                constraintsInfoPanel.weightx = 1.0;
                constraintsInfoPanel.weighty = 1.0;
                constraintsInfoPanel.insets = new  Insets(8, 44, 11, 36);
                add(getInfoPanel(), constraintsInfoPanel);

                GridBagConstraints constraintsModifyButtonsPanel = new  GridBagConstraints();
                constraintsModifyButtonsPanel.gridx = 1; constraintsModifyButtonsPanel.gridy = 3;
                constraintsModifyButtonsPanel.fill =  GridBagConstraints.BOTH;
                constraintsModifyButtonsPanel.weightx = 1.0;
                constraintsModifyButtonsPanel.weighty = 1.0;
                constraintsModifyButtonsPanel.insets = new  Insets(11, 109, 22, 101);
```

```
               add(getModifyButtonsPanel(), constraintsModifyButtonsPanel);

               GridBagConstraints constraintsHeadingLabel = new  GridBagConstraints();
               constraintsHeadingLabel.gridx = 1; constraintsHeadingLabel.gridy = 1;
               constraintsHeadingLabel.ipadx = 33;
               constraintsHeadingLabel.ipady = 10;
               constraintsHeadingLabel.insets = new  Insets(12, 117, 7, 110);
               add(getHeadingLabel(), constraintsHeadingLabel);
       } catch ( Throwable ivjExc) {
               handleException(ivjExc);
       }
       // user code begin {2}
       // user code end
}

/**
 * main entrypoint - starts the part when it is run as an application
 * @param args  String[]
 */
public static void main( String[] args) {
       try {
                JFrame frame = new  JFrame();
               UserFormPanel aUserFormPanel;
               aUserFormPanel = new UserFormPanel();
               frame.setContentPane(aUserFormPanel);
               frame.setSize(aUserFormPanel.getSize());
               frame.addWindowListener(new  WindowAdapter() {
                       public void windowClosing( WindowEvent e) {
                               System.exit(0);
                       };
               });
               frame.setVisible(true);
       } catch (Throwable exception) {
               System.err.println("Exception occurred in main() of  JPanel");
               exception.printStackTrace(System.out);
       }
}
}

/*
 * Revision History
 *
 */
```