Exercise 1 – Creating the ChatClient GUI Part 1 (Level 1)

Exercise objective – In this exercise, you will create a GUI for a "chat room" application. You will use a complex layout to properly position several GUI components in a frame.

Tasks

You will create a class called `ChatClient` that implements the following GUI design:

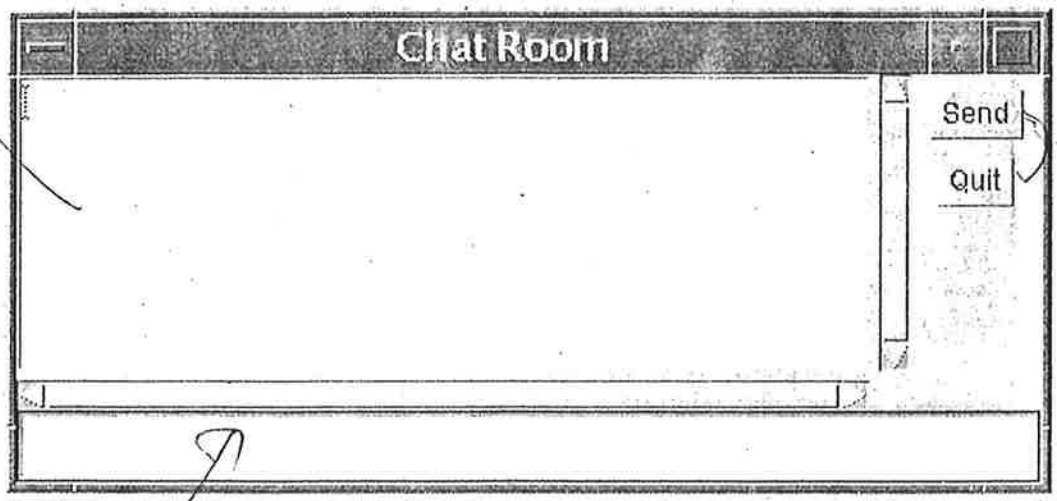


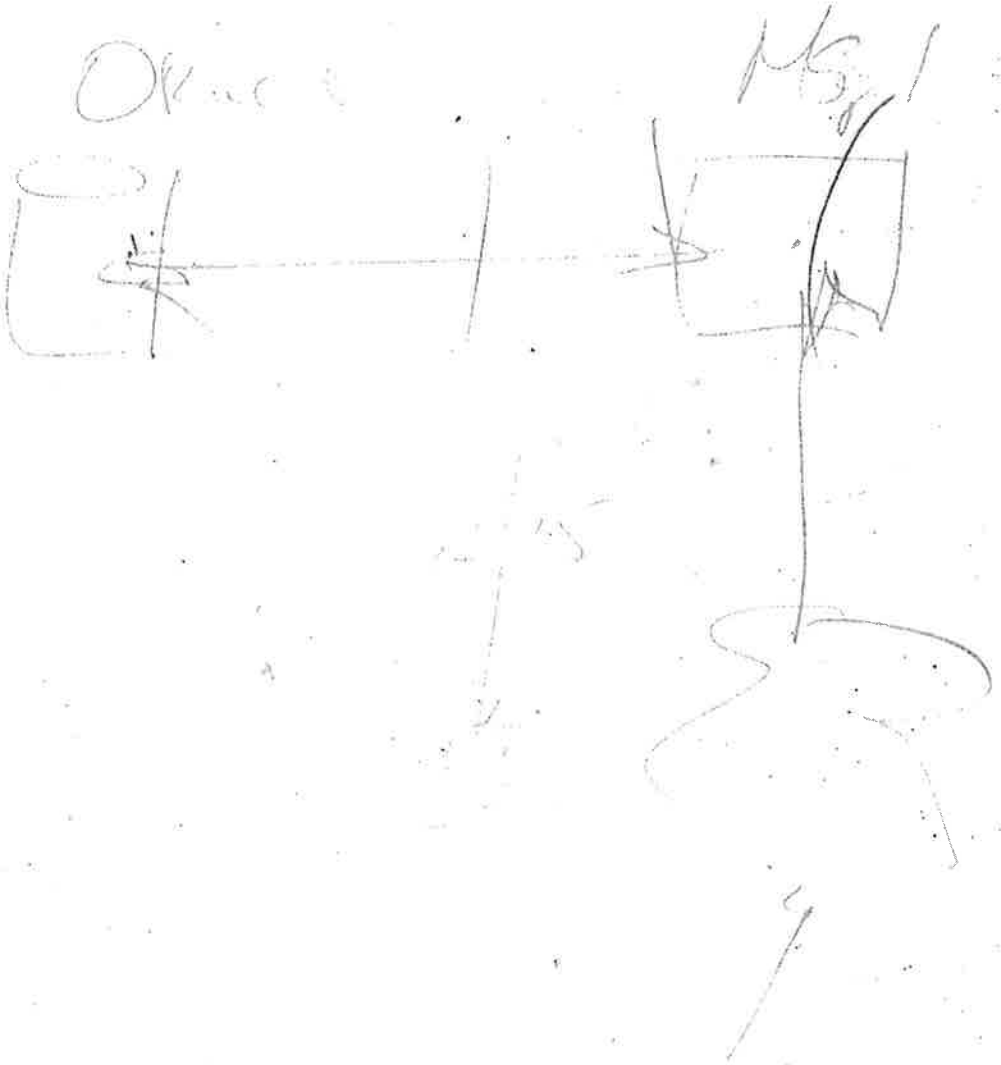
Figure 10-12 GUI Layout for the ChatClient Application

As you can see, there are four components in this GUI. The main component is a `TextArea`. The bottom component is a `TextField`. There are two `Button` components on the right. You will need to refer to Appendix C in the textbook for instructions on how to use the text components.

1. Create the `ChatClient` class with four private attributes; one for each component. In the constructor, initialize each of these component attributes: The text area should be 10 rows tall and 50 columns wide, the text field should be 50 columns wide, the send button should have the word "Send" in the display, and likewise for the quit button.

2. Create a `launchFrame` method which constructs the layout of the components. Feel free to use nested panels and any layout managers that will help you construct the layout in the GUI design shown above.

3. Create the `main` method. This method will instantiate a new `ChatClient` object and then call the `launchFrame` method.



10

Exercise 2 – Creating the Calculator GUI Part 1 (Level 2)

Exercise objective – In this exercise, you will create a GUI for a "calculator" application. You will use a grid layout to position the digit and operator buttons in a frame.

Tasks

You will create a class called `CalculatorGUI` that implements the following GUI design:

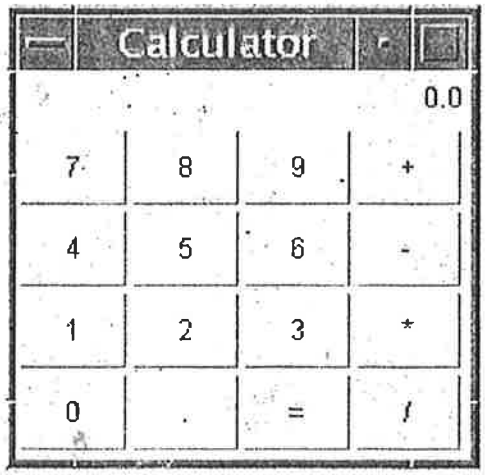
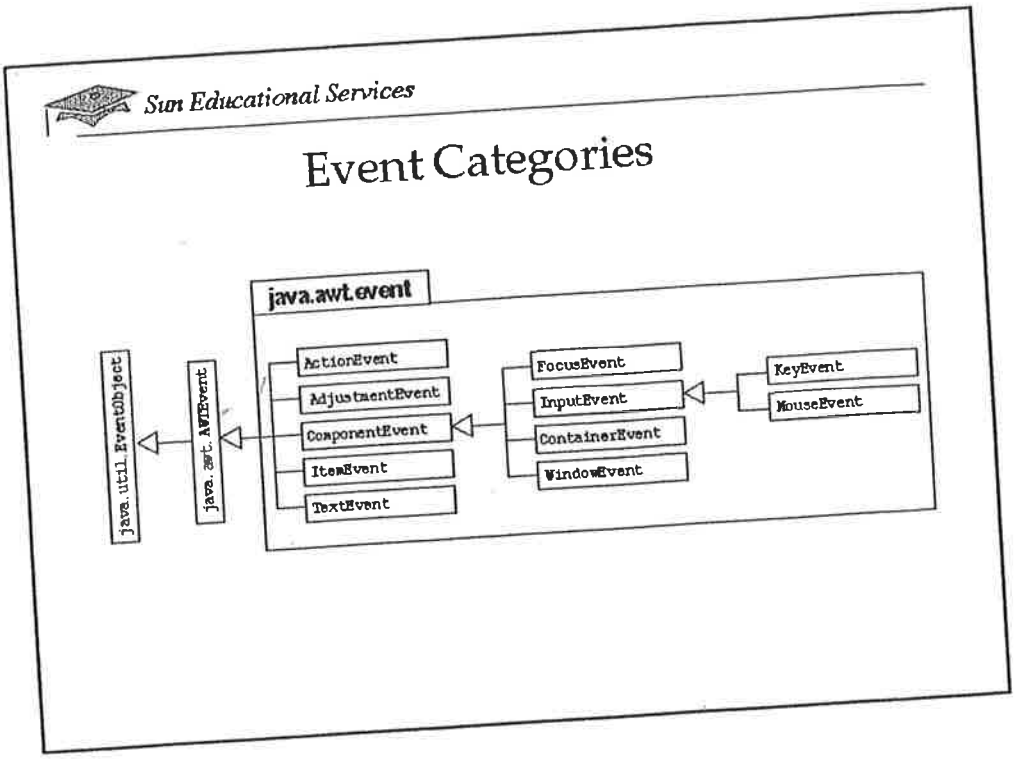


Figure 10-13 GUI Layout for the Calculator Application

Implement the `CalculatorGUI` class. As you can see there is a text field at the top of the frame (used to display the results) and a 4 by 4 grid of buttons. A label can also be used to display the results. Be creative!



GUI Behavior

This section describes the event categories.

Event Categories

The general mechanism for receiving events from components has been described in the context of a single type of event. Many of the event classes reside in the `java.awt.event` package, but others exist elsewhere in the API.

For each category of events, there is an interface that has to be implemented by the class of objects that wants to receive the events. That interface demands that one or more methods be defined as well. Those methods are called when particular events arise. Table 11-1 lists these categories and interfaces.

Table 11-1 Method Categories and Interfaces

Category	Interface Name	Methods
Action	`ActionListener`	`actionPerformed(ActionEvent)`
Item	`ItemListener`	`itemStateChanged(ItemEvent)`

Mouse	`MouseListener`	`mousePressed(MouseEvent)`
		`mouseReleased(MouseEvent)`
		`mouseEntered(MouseEvent)`
		`mouseExited(MouseEvent)`
		`mouseClicked(MouseEvent)`

Mouse Motion	`MouseMotionListener`	`mouseDragged(MouseEvent)` `mouseMoved(MouseEvent)`

Key	`KeyListener`	`keyPressed(KeyEvent)`
		`keyReleased(KeyEvent)`
		`keyTyped(KeyEvent)`

Focus	`FocusListener`	`focusGained(FocusEvent)` `focusLost(FocusEvent)`
Adjustment	`AdjustmentListener`	`adjustmentValueChanged (AdjustmentEvent)`

Component	`ComponentListener`	`componentMoved(ComponentEvent)`
		`componentHidden (ComponentEvent)`
		`componentResized(ComponentEvent)`
		`componentShown(ComponentEvent)`

Window	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`

Container	`ContainerListener`	`componentAdded(ContainerEvent)` `componentRemoved(ContainerEvent)`
Text	`TextListener`	`textValueChanged(TextEvent)`

11

Exercise 1 – Creating the ChatClient GUI, Part 2 (Level 2)

Exercise objective – In this exercise, you will implement the basic event handlers for the "chat room" GUI.

Tasks

Start by copying the `ChatClient` class file into your working directory.

1. Create an `ActionListener` which will copy the text from the input text field into the output text area when the send button is pressed. Use an inner class to implement this because you will need access to the ChatClient's private attributes.

2. Create an `ActionListener` which will quit the program when the quit button is pressed. [Hint: Use `System.exit(0)`]

3. Create a `WindowListener` which will quit the program when the close widget is pressed on the frame.

4. Create an `ActionListener` which will copy the text from the input text field into the output text area when the `<Enter>` key is pressed in the input text field.

5. Modify the `launchFrame` method to add instances of your listeners to the appropriate components.

Hints

1. Have the listeners (inner classes) access the instance variables of their containing class to refer to the components like the output text area and the input text field. Remember that you made the components instance variables in the previous lab.

2. Also, remember to import the `java.awt.event` package.

3. To get the text from a `TextArea` or `TextField`, you can use the `getText` method; to set the text use either the `setText` or append method.

Exercise 2 – Creating the Calculator GUI, Part 2 (Level 3)

Exercise objective – In this exercise, you will implement the basic event handlers for the "calculator" GUI.

Tasks

Start by coping the `CalculatorGUI` class file into your working directory.

Your task is to implement the `ActionListener` handlers for each of the buttons in the GUI. To make your job easier, you might want to copy the `Calculator` class in the solution directory. You can use an instance of this class as the model of "calculator" behavior. The behavior of this object is very simple. There is a method for each operation (`opAdd`, `opSubtract`, `opMultiply`, and `opDivide`) and the equals sign (`opEquals`). Each of these methods takes a single number (of type `String`) and returns a result (also of type `String`).

Here is an example (3.25 * 2 + 5 = 11.5):

```
Calculator calc = new Calculator();
String result;

result = calc.opMultiply("3.25"); // result is "3.25"
result = calc.opAdd("2");         // result is "6.5"
result = calc.opEquals("5");      // result is "11.5"
```

Using a model allows you to separate the logic of the application's behavior from the presentation and control of the application. This is a classic example of the design pattern of Model-View-Controller. In this case the `Calculator` class implements the model and the `CalculatorGUI` class implements both the view (the buttons, frame and layout) and the controller (the event handlers).

AWT Components

In Module 10, "Building Java GUIs," you were introduced to only one component (`Button`) and several containers (`Panel` and `Frame`). Table 12-1 briefly introduces you to the gamut of AWT components. For more information on the look and feel of these components, see Appendix C, "The AWT Component Library."

Table 12-1 AWT Component Descriptions

Component Type	Description
`Button`	A named rectangular box used for receiving mouse clicks.
`Canvas`	A panel used for drawing.
`Checkbox`	A component allowing the user to select an item.
`CheckboxMenuItem`	A checkbox within a menu.
`Choice`	A pull-down static list of items.
`Component`	The parent of all AWT components, except menu components.
`Container`	The parent of all AWT containers.
`Dialog`	A top-level window with a title and a border; dialogs can be modeless or modal.
`Frame`	The base class of all GUI windows with window manager controls.
`Label`	A text string component.
`List`	A component that contains a dynamic set of items.
`Menu`	An element under the menu bar, which contains a set of menu items.
`MenuItem`	An item within a menu.
`Panel`	A basic container class used most often to create complex layouts.
`Scrollbar`	A component that allows a user to "select from a range of values."
`ScrollPane`	A container class that implements automatic horizontal and vertical scrolling for a single child component.
`TextArea`	A component that allows the user to enter a block of text.

Table 12-1 AWT Component Descriptions (Continued)

Component Type	Description
`TextField`	A component that allows the user to enter a single line of text.
`Window`	The base class of all GUI windows, without window manager controls.

Component Events

Table 12-2 shows the basic AWT components and the event listeners that can be associated with that type of component.

Table 12-2 Components and Their Listeners

Component Type	Act	Adj	Cmp	Cnt	Foc	Itm	Key	Mou	MM	Text	Win
Button	✓		✓		✓		✓	✓	✓		
Canvas			✓		✓		✓	✓	✓		
Checkbox			✓		✓	✓	✓	✓	✓		
CheckboxMenuItem						✓					
Choice			✓		✓	✓	✓	✓	✓		
Component			✓		✓		✓	✓	✓		
Container			✓	✓	✓		✓	✓	✓		
Dialog			✓	✓	✓		✓	✓	✓		✓
Frame			✓	✓	✓		✓	✓	✓		✓
Label			✓		✓		✓	✓	✓		
List	✓		✓		✓	✓	✓	✓	✓		
MenuItem	✓										
Panel			✓	✓	✓		✓	✓	✓		
Scrollbar		✓	✓		✓		✓	✓	✓		
ScrollPane			✓	✓	✓		✓	✓	✓		
TextArea			✓		✓		✓	✓	✓	✓	
TextField	✓		✓		✓		✓	✓	✓	✓	
Window			✓	✓	✓		✓	✓	✓		✓

Act – ActionListener, **Adj** – AdjustmentListener,
Cmp – ComponentListener, **Cnt** – ContainerListener,
Foc – FocusListener, **Itm** – ItemListener, **Key** – KeyListener,
Mou – MouseListener, **MM** – MouseMotionListener,
Text – TextListener, **Win** – WindowListener