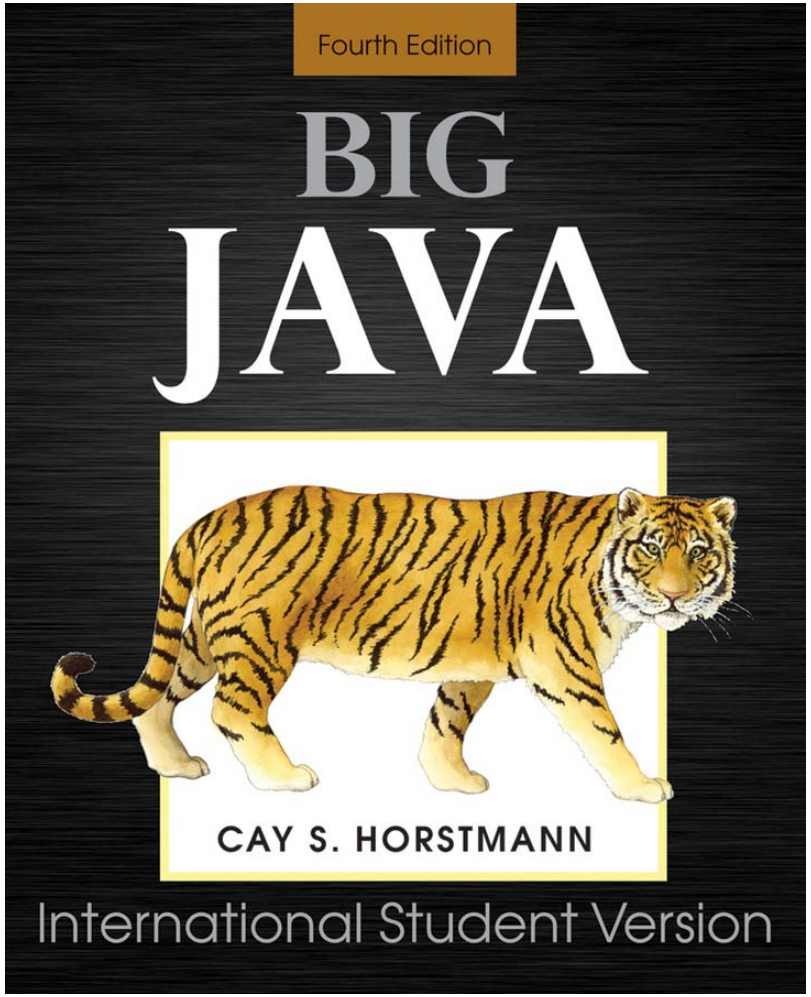


Chapter 17 – Graphical User Interfaces

Chapter Goals

- To become familiar with common user-interface components, such as buttons, combo boxes, text areas, and menus
- To understand the use of layout managers to arrange user-interface components in a container
- To build programs that handle events from user-interface components

Processing Text Input

- Use `JTextField` components to provide space for user input:

```
final int FIELD_WIDTH = 10; // In characters
final JTextField rateField = new JTextField(FIELD_WIDTH);
```

- Place a `JLabel` next to each text field:

```
JLabel rateLabel = new JLabel("Interest Rate: ");
```

Processing Text Input

- Supply a button that the user can press to indicate that the input is ready for processing:

Figure 1 An Application with a Text Field

Processing Text Input

- The `actionPerformed` method of the button's `ActionListener` reads the user input from the text fields (use `getText`):

```
class AddInterestListener implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      double rate =
         Double.parseDouble(rateField.getText());
      double interest = account.getBalance() * rate / 100;
      account.deposit(interest);
      resultLabel.setText("balance: ”
         + account.getBalance());
   }
}
```

ch17/textfield/InvestmentViewer3.java

```
1  import javax.swing.JFrame;
2
3  /**
4     This program displays the growth of an investment.
5  */
6  public class InvestmentViewer3
7  {
8     public static void main(String[] args)
9     {
10       JFrame frame = new InvestmentFrame();
11       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12       frame.setVisible(true);
13    }
14 }
```

ch17/textfield/InvestmentFrame.java

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5  import javax.swing.JLabel;
6  import javax.swing.JPanel;
7  import javax.swing.JTextField;
8
9  /**
10    A frame that shows the growth of an investment with variable interest.
11 */
12 public class InvestmentFrame extends JFrame
13 {
14    private static final int FRAME_WIDTH = 450;
15    private static final int FRAME_HEIGHT = 100;
16
17    private static final double DEFAULT_RATE = 5;
18    private static final double INITIAL_BALANCE = 1000;
19
```

Continued

ch17/textfield/InvestmentFrame.java (cont.)

```
20      private JLabel rateLabel;
21      private JTextField rateField;
22      private JButton button;
23      private JLabel resultLabel;
24      private JPanel panel;
25      private BankAccount account;
26
27      public InvestmentFrame()
28      {
29         account = new BankAccount(INITIAL_BALANCE);
30
31         // Use instance variables for components
32         resultLabel = new JLabel("balance: " + account.getBalance());
33
34         // Use helper methods
35         createTextField();
36         createButton();
37         createPanel();
38
39         setSize(FRAME_WIDTH, FRAME_HEIGHT);
40      }
```

Continued

ch17/textfield/InvestmentFrame.java (cont.)

```
42      private void createTextField()
43      {
44         rateLabel = new JLabel("Interest Rate: ");
45
46         final int FIELD_WIDTH = 10;
47         rateField = new JTextField(FIELD_WIDTH);
48         rateField.setText("" + DEFAULT_RATE);
49      }
50
```

Continued

ch17/textfield/InvestmentFrame.java (cont.)

```
51      private void createButton()
52      {
53         button = new JButton("Add Interest");
54
55         class AddInterestListener implements ActionListener
56         {
57            public void actionPerformed(ActionEvent event)
58            {
59               double rate = Double.parseDouble(rateField.getText());
60               double interest = account.getBalance() * rate / 100;
61               account.deposit(interest);
62               resultLabel.setText("balance: " + account.getBalance());
63            }
64         }
65
```

Continued

ch17/textfield/InvestmentFrame.java (cont.)

```
70      private void createPanel()
71      {
72         panel = new JPanel();
73         panel.add(rateLabel);
74         panel.add(rateField);
75         panel.add(button);
76         panel.add(resultLabel);
77         add(panel);
78      }
79   }
```

Self Check 17.1

What happens if you omit the first `JLabel` object?

Answer: Then the text field is not labeled, and the user will not know its purpose.

Self Check 17.2

If a text field holds an integer, what expression do you use to read its contents?

Answer: `Integer.parseInt(textField.getText())`

Text Areas

- Use a `JTextArea` to show multiple lines of text
- You can specify the number of rows and columns:

```
final int ROWS = 10;
final int COLUMNS = 30;
JTextArea textArea = new JTextArea(ROWS, COLUMNS);
```

- `setText`: to set the text of a text field or text area
- `append`: to add text to the end of a text area
- Use `newline` characters to separate lines:

```
textArea.append(account.getBalance() + "\n");
```

- To use for display purposes only:

```
textArea.setEditable(false);
// program can call setText and append to change it
```

Text Areas

- Classes `JTextField` and `JTextArea` are subclasses of library class `JTextComponent`
- Methods `setText` and `setEditable` are declared in the `JTextComponent` class and inherited by `JTextField` and `JTextArea`
- Method `append` is declared in the `JTextArea` class

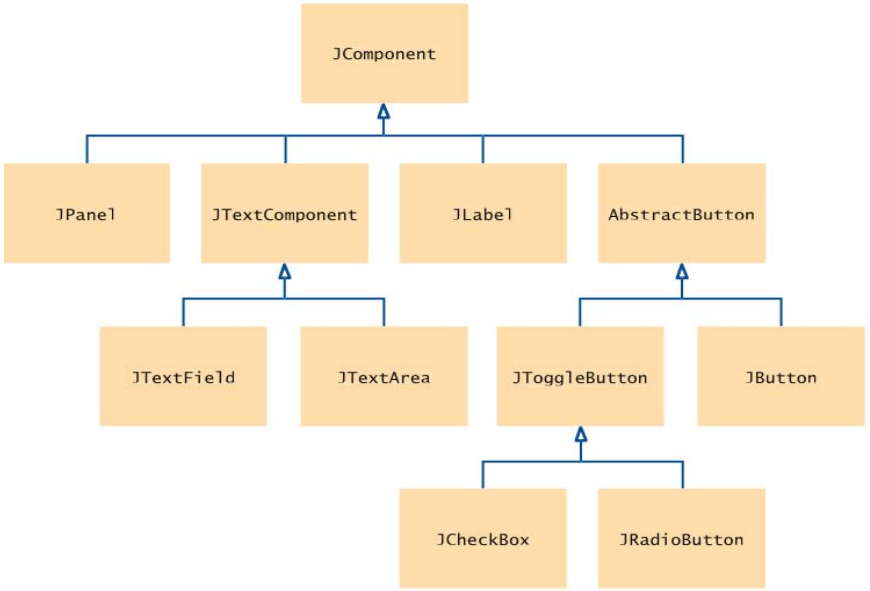


Figure 2 A Part of the Hierarchy of Swing User Interface Components

Text Areas

- To add scroll bars to a text area:

```
JTextArea textArea = new JTextArea(ROWS, COLUMNS);
JScrollPane scrollPane = new JScrollPane(textArea);
```

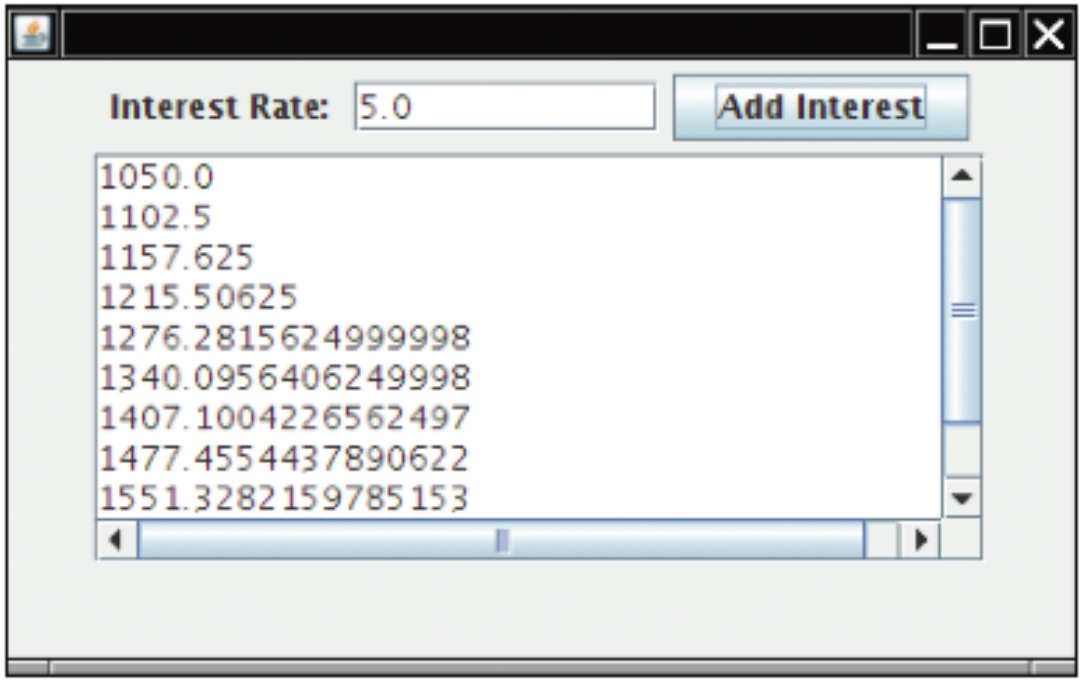


Figure 3 The Investment Application with a Text Area

ch17/textarea/InvestmentFrame.java

```
 1  import java.awt.event.ActionEvent;
 2  import java.awt.event.ActionListener;
 3  import javax.swing.JButton;
 4  import javax.swing.JFrame;
 5  import javax.swing.JLabel;
 6  import javax.swing.JPanel;
 7  import javax.swing.JScrollPane;
 8  import javax.swing.JTextArea;
 9  import javax.swing.JTextField;
10
11  /**
12     A frame that shows the growth of an investment with variable interest.
13  */
14  public class InvestmentFrame extends JFrame
15  {
16     private static final int FRAME_WIDTH = 400;
17     private static final int FRAME_HEIGHT = 250;
18
19     private static final int AREA_ROWS = 10;
20     private static final int AREA_COLUMNS = 30;
21     private static final double DEFAULT_RATE = 5;
22     private static final double INITIAL_BALANCE = 1000;
```

Continued

ch17/textarea/InvestmentFrame.java (cont.)

```
23
24      private JLabel rateLabel;
25      private JTextField rateField;
26      private JButton button;
27      private JTextArea resultArea;
28      private JPanel panel;
29      private BankAccount account;
30
31      public InvestmentFrame()
32      {
33         account = new BankAccount(INITIAL_BALANCE);
34         resultArea = new JTextArea(AREA_ROWS, AREA_COLUMNS);
35         resultArea.setEditable(false);
36
37         // Use helper methods
38         createTextField();
39         createButton();
40         createPanel();
41
42         setSize(FRAME_WIDTH, FRAME_HEIGHT);
43      }
44
```

Continued

ch17/textarea/InvestmentFrame.java (cont.)

```
45      private void createTextField()
46      {
47         rateLabel = new JLabel("Interest Rate: ");
48
49         final int FIELD_WIDTH = 10;
50         rateField = new JTextField(FIELD_WIDTH);
51         rateField.setText("" + DEFAULT_RATE);
52      }
53
54      private void createButton()
55      {
56         button = new JButton("Add Interest");
57
58         class AddInterestListener implements ActionListener
59         {
60            public void actionPerformed(ActionEvent event)
61            {
62               double rate = Double.parseDouble(rateField.getText());
63               double interest = account.getBalance() * rate / 100;
64               account.deposit(interest);
65               resultArea.append(account.getBalance() + "\n");
66            }
67         }
68
```

Continued

ch17/textarea/InvestmentFrame.java (cont.)

```
69          ActionListener listener = new AddInterestListener();
70          button.addActionListener(listener);
71       }
72
73       private void createPanel()
74       {
75          panel = new JPanel();
76          panel.add(rateLabel);
77          panel.add(rateField);
78          panel.add(button);
79          JScrollPane scrollPane = new JScrollPane(resultArea);
80          panel.add(scrollPane);
81          add(panel);
82       }
83    }
```

Self Check 17.3

What is the difference between a text field and a text area?

Answer: A text field holds a single line of text; a text area holds multiple lines.

Self Check 17.4

Why did the `InvestmentFrame` program call `resultArea.setEditable(false)`?

Answer: The text area is intended to display the program output. It does not collect user input.

Self Check 17.5

How would you modify the `InvestmentFrame` program if you didn't want to use scroll bars?

Answer: Don't construct a `JScrollPane` and add the `resultArea` object directly to the frame.

Layout Management

- Up to now, we have had limited control over layout of components
  - *When we used a panel, it arranged the components from the left to the right*
- User-interface components are arranged by placing them inside containers
  - *Containers can be placed inside larger containers*
- Each container has a *layout manager* that directs the arrangement of its components
- Three useful layout managers:
  - *border layout*
  - *flow layout*
  - *grid layout*

Layout Management

- By default, `JPanel` places components from left to right and starts a new row when needed
- Panel layout carried out by `FlowLayout` layout manager
- Can set other layout managers:

```
panel.setLayout(new BorderLayout());
```

Border Layout

- Border layout groups container into five areas - center, north, west, south and east

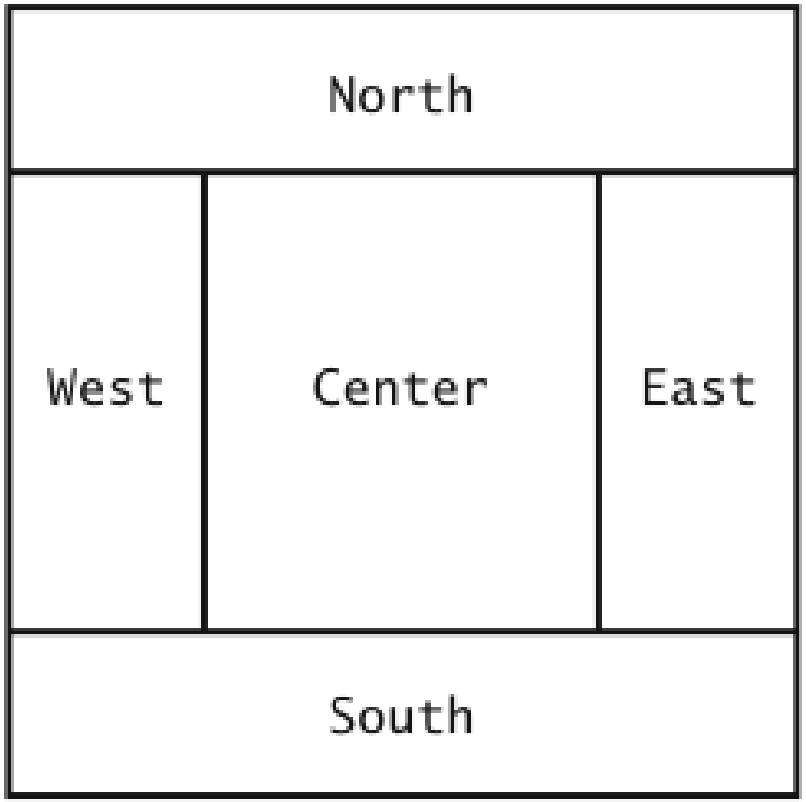


Figure 4
Components Expand to Fill Space in the Border Layout

Border Layout

- Default layout manager for a frame (technically, the frame's content pane)
- When adding a component, specify the position like this:

```
panel.add(component, BorderLayout.NORTH);
```

- Expands each component to fill the entire allotted area
  If that is not desirable, place each component inside a panel

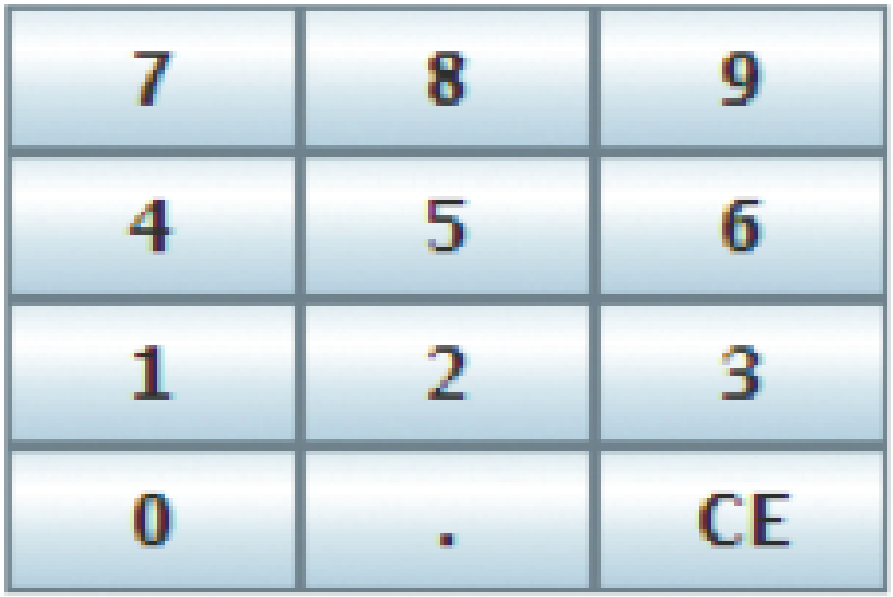
Grid Layout

- Arranges components in a grid with a fixed number of rows and columns
- Resizes each component so that they all have same size
- Expands each component to fill the entire allotted area
- Add the components, row by row, left to right:

```
JPanel numberPanel = new JPanel();
numberPanel.setLayout(new GridLayout(4, 3));
numberPanel.add(button7);
numberPanel.add(button8);
numberPanel.add(button9);
numberPanel.add(button4);
...
```

Grid Layout

Figure 5
The Grid Layout



Grid Bag Layout

- Tabular arrangement of components
  - *Columns can have different sizes*
  - *Components can span multiple columns*
- Quite complex to use
- Not covered in the book
- Fortunately, you can create acceptable-looking layouts by nesting panels
  - *Give each panel an appropriate layout manager*
  - *Panels don't have visible borders*
  - *Use as many panels as needed to organize components*

Nesting Panels Example

Keypad from the ATM GUI in Chapter 12:

```
JPanel keypadPanel = new JPanel();
keypadPanel.setLayout(new BorderLayout());
buttonPanel = new JPanel();
buttonPanel.setLayout(new GridLayout(4, 3));
buttonPanel.add(button7);
buttonPanel.add(button8);
// ...
keypadPanel.add(buttonPanel, BorderLayout.CENTER);
 JTextField display = new JTextField();
keypadPanel.add(display, BorderLayout.NORTH);
```

Nesting Panels Example

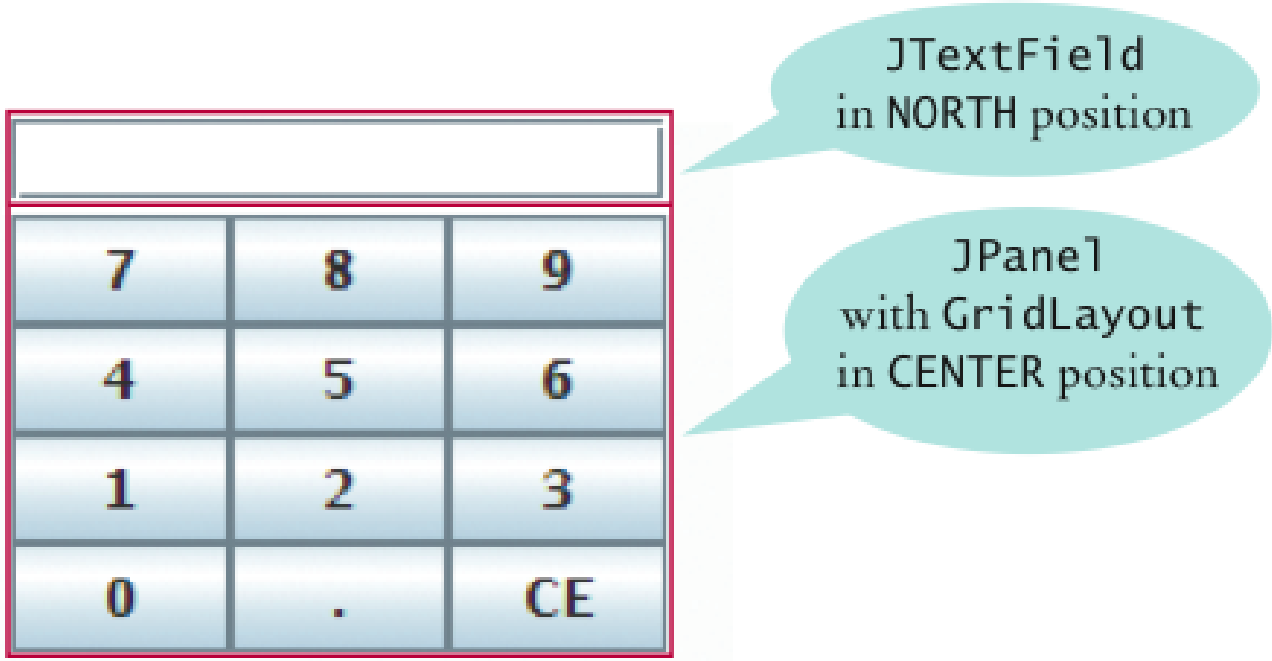


Figure 6
Nesting Panels

Self Check 17.6

How do you add two buttons to the north area of a frame?

Answer: First add them to a panel, then add the panel to the north end of a frame.

Self Check 17.7

How can you stack three buttons on top of each other?

Answer: Place them inside a panel with a `GridLayout` that has three rows and one column.

Choices

- Radio buttons
- Check boxes
- Combo boxes

Figure 7
A Combo Box, Check Boxes, and Radio Buttons

Radio Buttons

- For a small set of mutually exclusive choices, use radio buttons or a combo box
- In a radio button set, only one button can be selected at a time
- When a button is selected, previously selected button in set is automatically turned off

Radio Buttons

- In previous figure, font sizes are mutually exclusive:

```
JRadioButton smallButton = new JRadioButton("Small");
JRadioButton mediumButton = new JRadioButton("Medium");
JRadioButton largeButton = new JRadioButton("Large");

// Add radio buttons into a ButtonGroup so that
// only one button in group is on at any time
ButtonGroup group = new ButtonGroup();
group.add(smallButton);
group.add(mediumButton);
group.add(largeButton);
```

Radio Buttons

- Button group does not place buttons close to each other on container
- It is your job to arrange buttons on screen
- `isSelected`: Called to find out if a button is currently selected or not:

```
if(largeButton.isSelected()) size = LARGE_SIZE
```

- Call `setSelected(true)` on a radio button in group before making the enclosing frame visible

Borders

- Place a border around a panel to group its contents visually
- `EtchedBorder`: Three-dimensional etched effect
- Can add a border to any component, but most commonly to panels:

```
JPanel panel = new JPanel();
panel.setBorder(new EtchedBorder());
```

- `TitledBorder`: A border with a title:

```
panel.setBorder(new TitledBorder(new EtchedBorder(),
   "Size"));
```

Check Boxes

- Two states: Checked and unchecked
- Use one checkbox for a binary choice
- Use a group of check boxes when one selection does not exclude another
- Example: “Bold” and “Italic” in previous figure
- Construct by giving the name in the constructor:

```
JCheckBox italicCheckBox = new JCheckBox("Italic");
```

- Don’t place into a button group

Combo Boxes

- For a large set of choices, use a combo box
  - *Uses less space than radio buttons*
- "Combo": Combination of a list and a text field
  - *The text field displays the name of the current selection*

Figure 8
An Open Combo Box

Combo Boxes

- If combo box is editable, user can type own selection
  - *Use* `setEditable` *method*
- Add strings with `addItem` method:

```
JComboBox facenameCombo = new JComboBox();
facenameCombo.addItem("Serif");
facenameCombo.addItem("SansSerif");
...
```

- Get user selection with `getSelectedItem` (return type is `Object`):

```
String selectedString =
(String) facenameCombo.getSelectedItem();
```

- Select an item with `setSelectedItem`

Radio Buttons, Check Boxes, and Combo Boxes

- They generate an `ActionEvent` whenever the user selects an item
- An example: `FontViewerFrame`

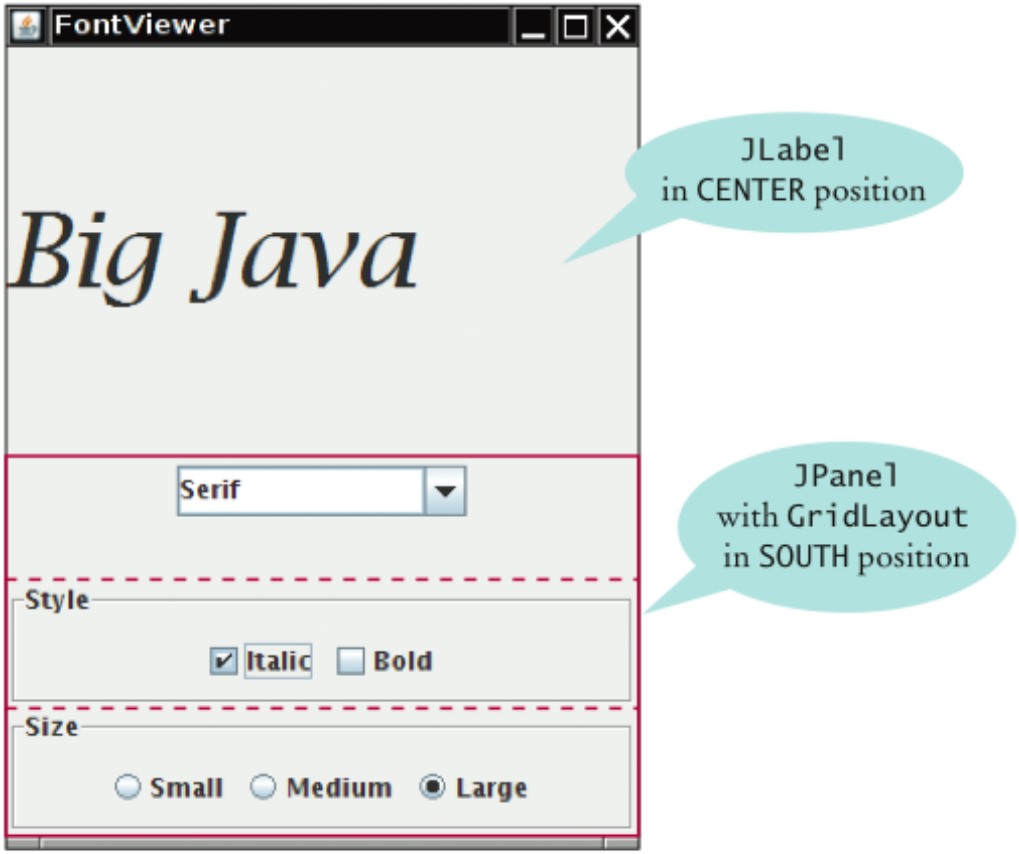


Figure 9 The Components of the `FontViewerFrame`

Radio Buttons, Check Boxes, and Combo Boxes

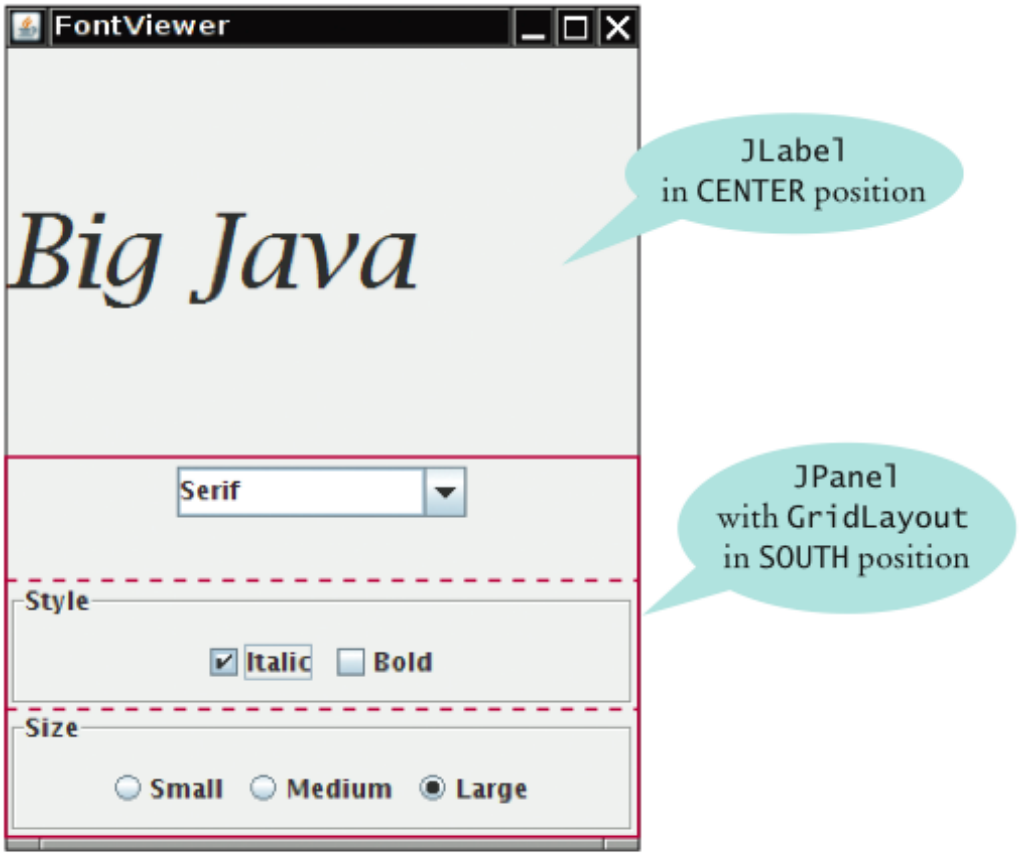


Figure 9 The Components of the `FontViewerFrame`

- All components notify the same listener object
- When user clicks on any component, we ask each component for its current content
- Then redraw text sample with the new font

Classes of the Font Viewer Program

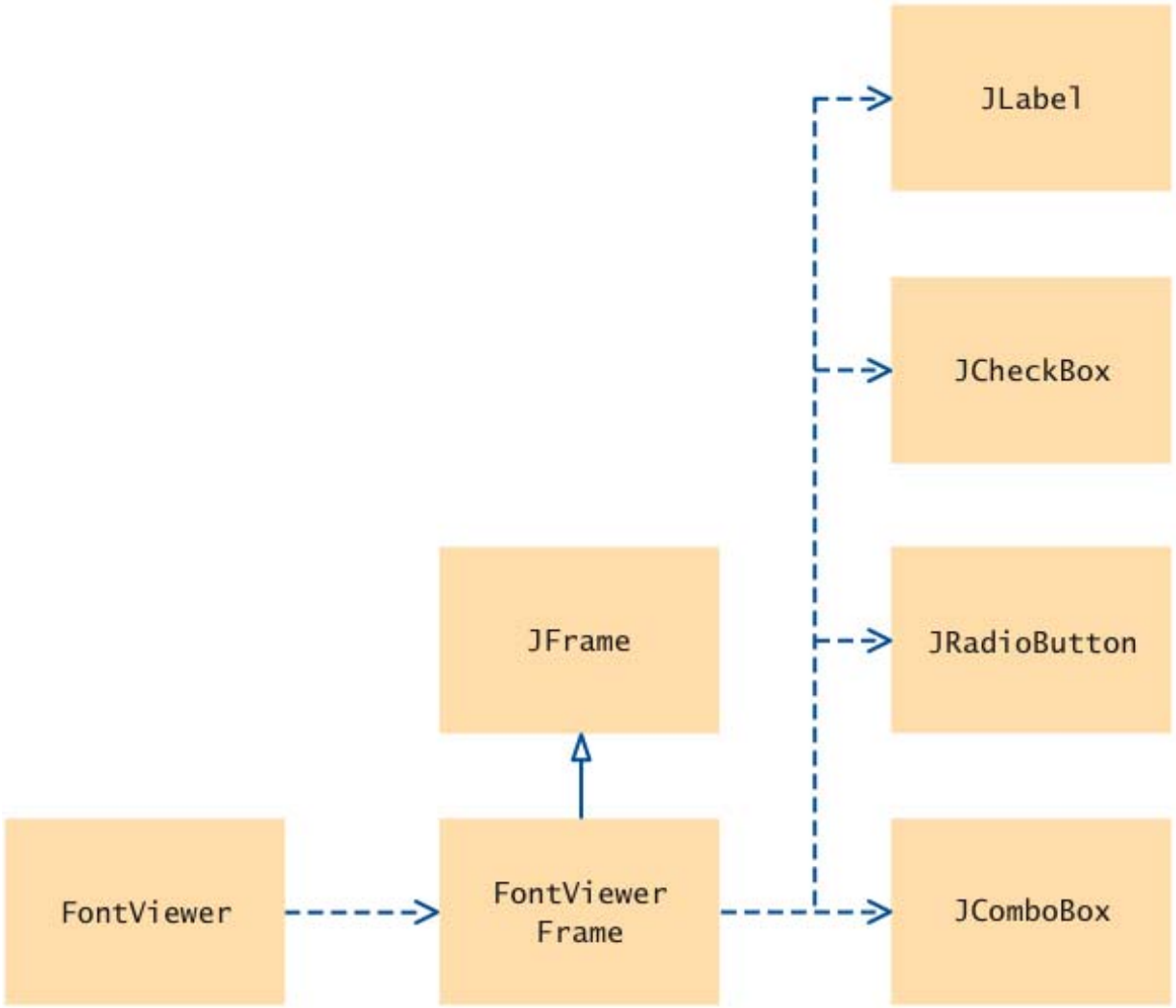


Figure 10 Classes of the Font Viewer Program

ch17/choice/FontViewer.java

```
1  import javax.swing.JFrame;
2
3  /**
4     This program allows the user to view font effects.
5  */
6  public class FontViewer
7  {
8     public static void main(String[] args)
9     {
10       JFrame frame = new FontViewerFrame();
11       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12       frame.setTitle("FontViewer");
13       frame.setVisible(true);
14    }
15 }
```

ch17/choice/FontViewerFrame.java

```
 1  import java.awt.BorderLayout;
 2  import java.awt.Font;
 3  import java.awt.GridLayout;
 4  import java.awt.event.ActionEvent;
 5  import java.awt.event.ActionListener;
 6  import javax.swing.ButtonGroup;
 7  import javax.swing.JButton;
 8  import javax.swing.JCheckBox;
 9  import javax.swing.JComboBox;
10  import javax.swing.JFrame;
11  import javax.swing.JLabel;
12  import javax.swing.JPanel;
13  import javax.swing.JRadioButton;
14  import javax.swing.border.EtchedBorder;
15  import javax.swing.border.TitledBorder;
16
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
17  /**
18     This frame contains a text field and a control panel
19     to change the font of the text.
20  */
21  public class FontViewerFrame extends JFrame
22  {
23     private static final int FRAME_WIDTH = 300;
24     private static final int FRAME_HEIGHT = 400;
25
26     private JLabel sampleField;
27     private JCheckBox italicCheckBox;
28     private JCheckBox boldCheckBox;
29     private JRadioButton smallButton;
30     private JRadioButton mediumButton;
31     private JRadioButton largeButton;
32     private JComboBox facenameCombo;
33     private ActionListener listener;
34
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
35      /**
36         Constructs the frame.
37      */
38      public FontViewerFrame()
39      {
40         // Construct text sample
41         sampleField = new JLabel("Big Java");
42         add(sampleField, BorderLayout.CENTER);
43
44         // This listener is shared among all components
45         class ChoiceListener implements ActionListener
46         {
47            public void actionPerformed(ActionEvent event)
48            {
49               setSampleFont();
50            }
51         }
52
53         listener = new ChoiceListener();
54
55         createControlPanel();
56         setSampleFont();
57         setSize(FRAME_WIDTH, FRAME_HEIGHT);
58      }
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
60      /**
61         Creates the control panel to change the font.
62      */
63      public void createControlPanel()
64      {
65         JPanel facenamePanel = createComboBox();
66         JPanel sizeGroupPanel = createCheckBoxes();
67         JPanel styleGroupPanel = createRadioButtons();
68
69         // Line up component panels
70
71         JPanel controlPanel = new JPanel();
72         controlPanel.setLayout(new GridLayout(3, 1));
73         controlPanel.add(facenamePanel);
74         controlPanel.add(sizeGroupPanel);
75         controlPanel.add(styleGroupPanel);
76
77         // Add panels to content pane
78
79         add(controlPanel, BorderLayout.SOUTH);
80      }
81
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
82      /**
83         Creates the combo box with the font style choices.
84         @return the panel containing the combo box
85      */
86      public JPanel createComboBox()
87      {
88         facenameCombo = new JComboBox();
89         facenameCombo.addItem("Serif");
90         facenameCombo.addItem("SansSerif");
91         facenameCombo.addItem("Monospaced");
92         facenameCombo.setEditable(true);
93         facenameCombo.addActionListener(listener);
94
95         JPanel panel = new JPanel();
96         panel.add(facenameCombo);
97         return panel;
98      }
99
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
100      /**
101         Creates the check boxes for selecting bold and italic styles.
102         @return the panel containing the check boxes
103      */
104      public JPanel createCheckBoxes()
105      {
106         italicCheckBox = new JCheckBox("Italic");
107         italicCheckBox.addActionListener(listener);
108
109         boldCheckBox = new JCheckBox("Bold");
110         boldCheckBox.addActionListener(listener);
111
112         JPanel panel = new JPanel();
113         panel.add(italicCheckBox);
114         panel.add(boldCheckBox);
115         panel.setBorder(new TitledBorder(new EtchedBorder(),
"Style"));
116
117         return panel;
118      }
119
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
120     /**
121        Creates the radio buttons to select the font size
122        @return the panel containing the radio buttons
123     */
124     public JPanel createRadioButtons()
125     {
126        smallButton = new JRadioButton("Small");
127        smallButton.addActionListener(listener);
128
129        mediumButton = new JRadioButton("Medium");
130        mediumButton.addActionListener(listener);
131
132        largeButton = new JRadioButton("Large");
133        largeButton.addActionListener(listener);
134        largeButton.setSelected(true);
135
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
136         // Add radio buttons to button group
137
138         ButtonGroup group = new ButtonGroup();
139         group.add(smallButton);
140         group.add(mediumButton);
141         group.add(largeButton);
142
143         JPanel panel = new JPanel();
144         panel.add(smallButton);
145         panel.add(mediumButton);
146         panel.add(largeButton);
147         panel.setBorder(new TitledBorder(new EtchedBorder(), "Size"));
148
149         return panel;
150      }
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
152    /**
153       Gets user choice for font name, style, and size
154       and sets the font of the text sample.
155    */
156    public void setSampleFont()
157    {
158       // Get font name
159       String facename
160             = (String) facenameCombo.getSelectedItem();
161
162       // Get font style
163
164       int style = 0;
165       if (italicCheckBox.isSelected())
166       {
167          style = style + Font.ITALIC;
168       }
169       if (boldCheckBox.isSelected())
170       {
171          style = style + Font.BOLD;
172       }
173
```

Continued

ch17/choice/FontViewerFrame.java (cont.)

```
174         // Get font size
175
176         int size = 0;
177
178         final int SMALL_SIZE = 24;
179         final int MEDIUM_SIZE = 36;
180         final int LARGE_SIZE = 48;
181
182         if (smallButton.isSelected()) { size = SMALL_SIZE; }
183         else if (mediumButton.isSelected()) { size = MEDIUM_SIZE; }
184         else if (largeButton.isSelected()) { size = LARGE_SIZE; }
185
186         // Set font of text field
187
188         sampleField.setFont(new Font(facename, style, size));
189         sampleField.repaint();
190      }
191   }
```

Continued

Self Check 17.8

What is the advantage of a `JComboBox` over a set of radio buttons? What is the disadvantage?

Answer: If you have many options, a set of radio buttons takes up a large area. A combo box can show many options without using up much space. But the user cannot see the options as easily.

Self Check 17.9

Why do all user interface components in the `FontViewerFrame` class share the same listener?

Answer: When any of the component settings is changed, the program simply queries all of them and updates the label.

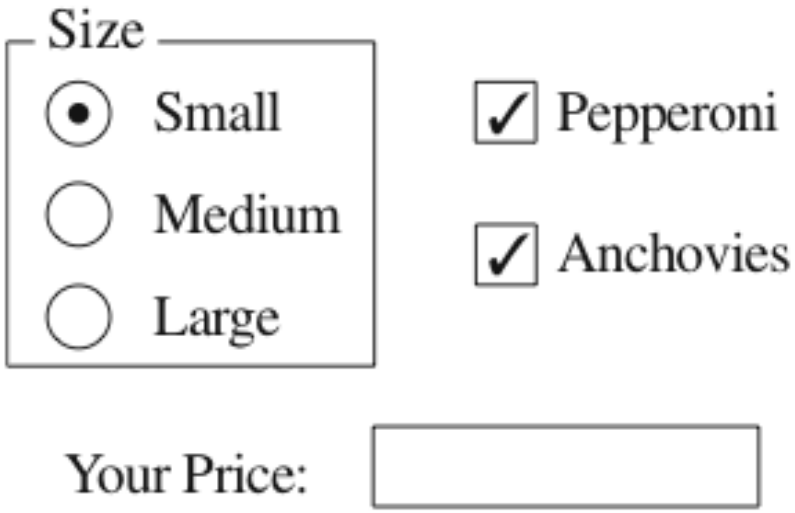
Self Check 17.10

Why was the combo box placed inside a panel? What would have happened if it had been added directly to the control panel?

Answer: To keep it from growing too large. It would have grown to the same width and height as the two panels below it.

How To 17.1 Laying Out a User Interface

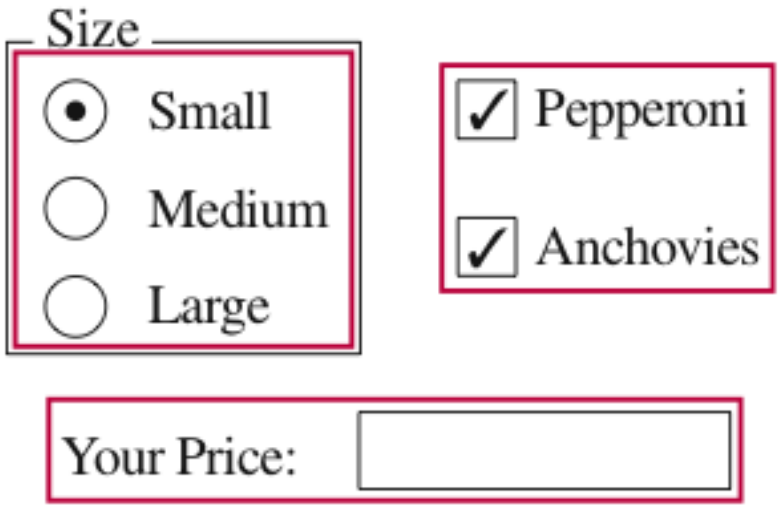
Step 1: Make a sketch of your desired component layout



Continued

How To 17.1 Laying Out a User Interface (cont.)

Step 2: Find groupings of adjacent components with the same layout

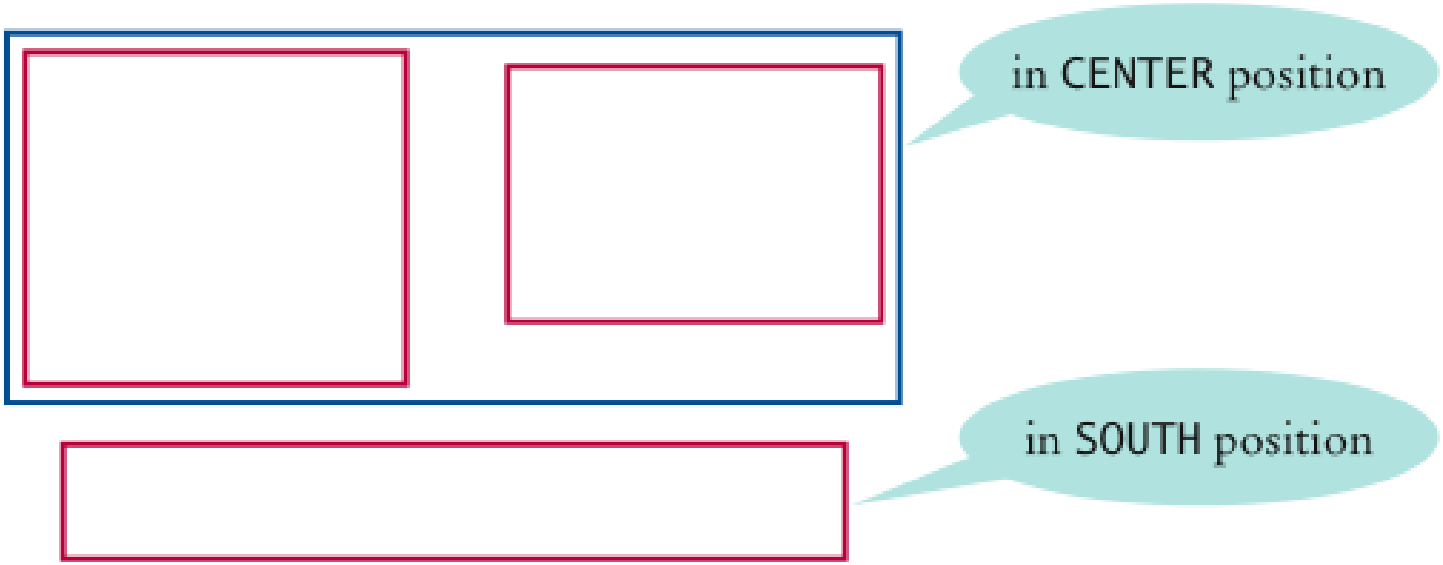


Continued

How To 17.1 Laying Out a User Interface (cont.)

Step 3: Identify layouts for each group

Step 4: Group the groups together



Step 5: Write the code to generate the layout

GUI Builder

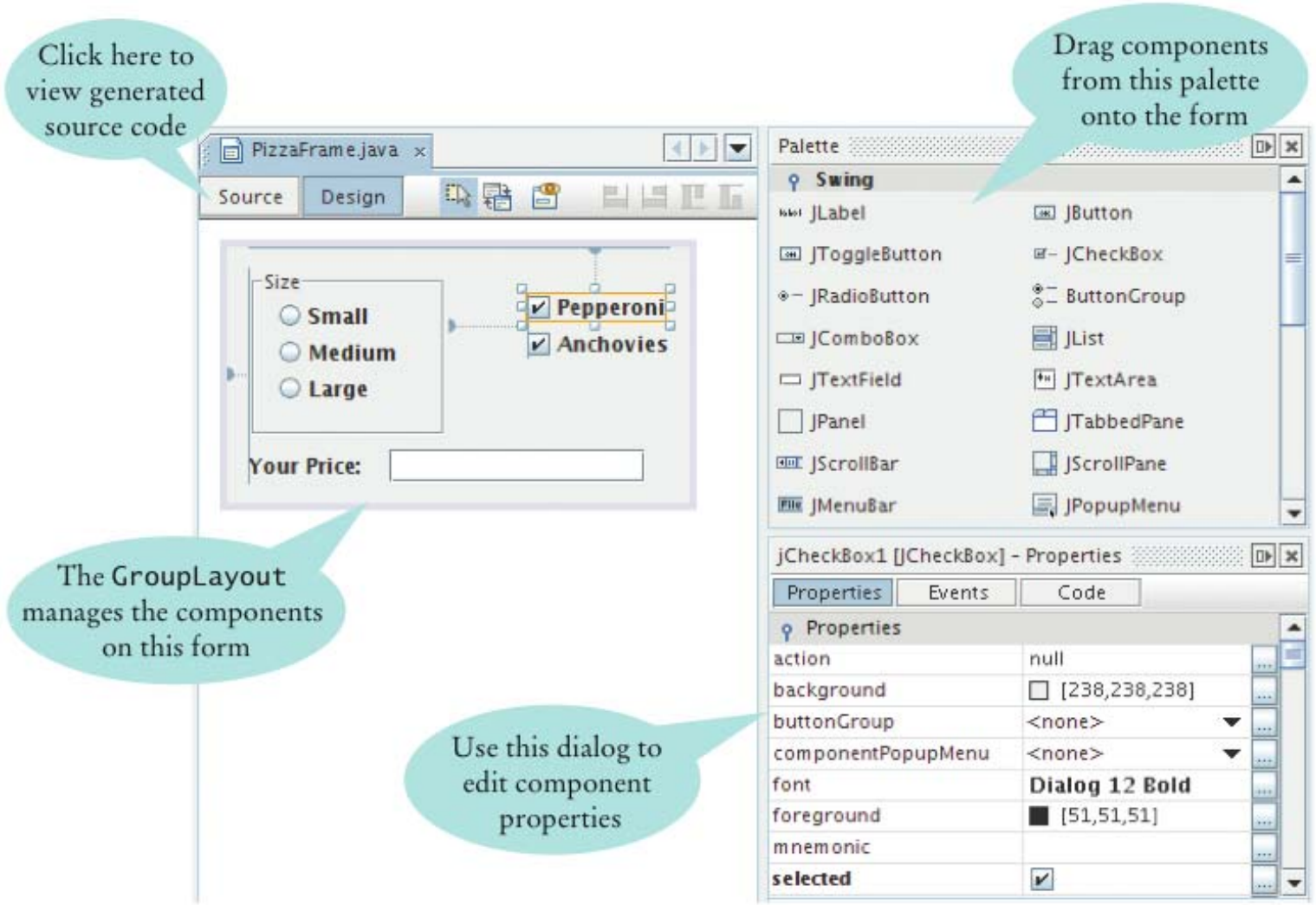
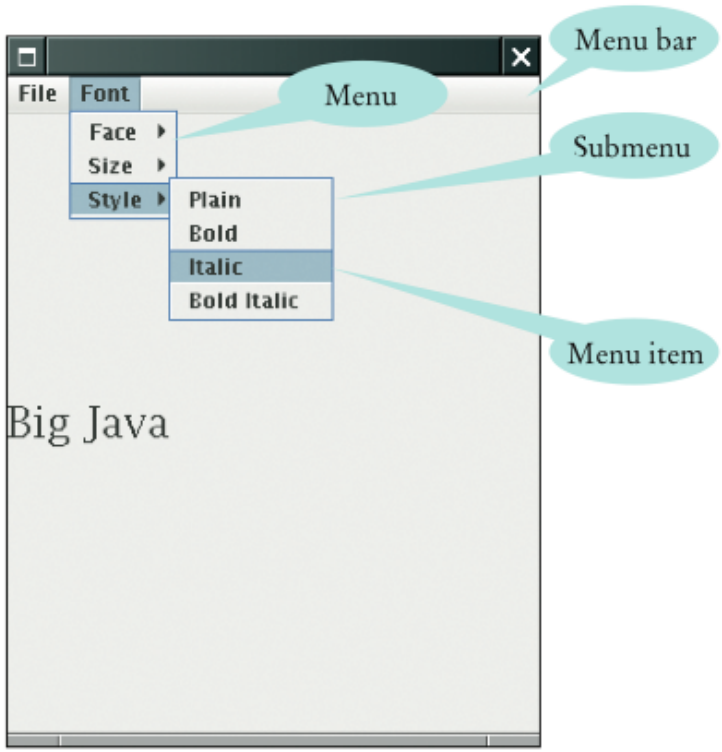


Figure 11 A GUI Builder

Menus

- A frame contains a menu bar
- The menu bar contains menus
- A menu contains submenus and menu items

Figure 12
Pull-Down Menus



Menu Items

- Add menu items and submenus with the `add` method:

```
JMenuItem fileExitItem = new JMenuItem("Exit");
fileMenu.add(fileExitItem);
```

- A menu item has no further submenus
- Menu items generate action events
- Add a listener to each menu item:

```
fileExitItem.addActionListener(listener);
```

- Add action listeners only to menu items, not to menus or the menu bar

A Sample Program

- Builds up a small but typical menu
- Traps action events from menu items
- To keep program readable, use a separate method for each menu or set of related menus
  - `createFaceItem`: *creates menu item to change the font face*
  - `createSizeItem`
  - `createStyleItem`

ch17/menu/FontViewer2.java

```
1  import javax.swing.JFrame;
2
3  /**
4     This program uses a menu to display font effects.
5  */
6  public class FontViewer2
7  {
8     public static void main(String[] args)
9     {
10       JFrame frame = new FontViewer2Frame();
11       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12       frame.setVisible(true);
13    }
14 }
15
```

ch17/menu/FontViewer2Frame.java

```
 1  import java.awt.BorderLayout;
 2  import java.awt.Font;
 3  import java.awt.GridLayout;
 4  import java.awt.event.ActionEvent;
 5  import java.awt.event.ActionListener;
 6  import javax.swing.ButtonGroup;
 7  import javax.swing.JButton;
 8  import javax.swing.JCheckBox;
 9  import javax.swing.JComboBox;
10  import javax.swing.JFrame;
11  import javax.swing.JLabel;
12  import javax.swing.JMenu;
13  import javax.swing.JMenuBar;
14  import javax.swing.JMenuItem;
15  import javax.swing.JPanel;
16  import javax.swing.JRadioButton;
17  import javax.swing.border.EtchedBorder;
18  import javax.swing.border.TitledBorder;
19
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
20  /**
21     This frame has a menu with commands to change the font
22     of a text sample.
23  */
24  public class FontViewer2Frame extends JFrame
25  {
26     private static final int FRAME_WIDTH = 300;
27     private static final int FRAME_HEIGHT = 400;
28
29     private JLabel sampleField;
30     private String facename;
31     private int fontstyle;
32     private int fontsize;
33
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
34      /**
35         Constructs the frame.
36      */
37      public FontViewer2Frame()
38      {
39         // Construct text sample
40         sampleField = new JLabel("Big Java");
41         add(sampleField, BorderLayout.CENTER);
42
43         // Construct menu
44         JMenuBar menuBar = new JMenuBar();
45         setJMenuBar(menuBar);
46         menuBar.add(createFileMenu());
47         menuBar.add(createFontMenu());
48
49         facename = "Serif";
50         fontsize = 24;
51         fontstyle = Font.PLAIN;
52
53         setSampleFont();
54         setSize(FRAME_WIDTH, FRAME_HEIGHT);
55      }
56
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
57      /**
58         Creates the File menu.
59         @return the menu
60      */
61      public JMenu createFileMenu()
62
63      {
64         JMenu menu = new JMenu("File");
65         menu.add(createFileExitItem());
66         return menu;
67      }
68
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
69      /**
70         Creates the File->Exit menu item and sets its action listener.
71         @return the menu item
72      */
73      public JMenuItem createFileExitItem()
74      {
75         JMenuItem item = new JMenuItem("Exit");
76         class MenuItemListener implements ActionListener
77         {
78            public void actionPerformed(ActionEvent event)
79            {
80               System.exit(0);
81            }
82         }
83         ActionListener listener = new MenuItemListener();
84         item.addActionListener(listener);
85         return item;
86      }
87
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
88      /**
89         Creates the Font submenu.
90         @return the menu
91      */
92      public JMenu createFontMenu()
93      {
94         JMenu menu = new JMenu("Font");
95         menu.add(createFaceMenu());
96         menu.add(createSizeMenu());
97         menu.add(createStyleMenu());
98         return menu;
99      }
100
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
101      /**
102         Creates the Face submenu.
103         @return the menu
104      */
105      public JMenu createFaceMenu()
106      {
107         JMenu menu = new JMenu("Face");
108         menu.add(createFaceItem("Serif"));
109         menu.add(createFaceItem("SansSerif"));
110         menu.add(createFaceItem("Monospaced"));
111         return menu;
112      }
113
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
114      /**
115         Creates the Size submenu.
116         @return the menu
117      */
118      public JMenu createSizeMenu()
119      {
120         JMenu menu = new JMenu("Size");
121         menu.add(createSizeItem("Smaller", -1));
122         menu.add(createSizeItem("Larger", 1));
123         return menu;
124      }
125
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
126     /**
127        Creates the Style submenu.
128        @return the menu
129     */
130     public JMenu createStyleMenu()
131     {
132        JMenu menu = new JMenu("Style");
133        menu.add(createStyleItem("Plain", Font.PLAIN));
134        menu.add(createStyleItem("Bold", Font.BOLD));
135        menu.add(createStyleItem("Italic", Font.ITALIC));
136        menu.add(createStyleItem("Bold Italic", Font.BOLD
137              + Font.ITALIC));
138        return menu;
139     }
140
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
141     /**
142        Creates a menu item to change the font face and set its action listener.
143        @param name the name of the font face
144        @return the menu item
145     */
146     public JMenuItem createFaceItem(final String name)
147     {
148        JMenuItem item = new JMenuItem(name);
149        class MenuItemListener implements ActionListener
150        {
151           public void actionPerformed(ActionEvent event)
152           {
153              facename = name;
154              setSampleFont();
155           }
156        }
157        ActionListener listener = new MenuItemListener();
158        item.addActionListener(listener);
159        return item;
160     }
161
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
162     /**
163        Creates a menu item to change the font size
164        and set its action listener.
165        @param name the name of the menu item
166        @param ds the amount by which to change the size
167        @return the menu item
168     */
169     public JMenuItem createSizeItem(String name, final int ds)
170     {
171        JMenuItem item = new JMenuItem(name);
172        class MenuItemListener implements ActionListener
173        {
174           public void actionPerformed(ActionEvent event)
175           {
176              fontsize = fontsize + ds;
177              setSampleFont();
178           }
179        }
180        ActionListener listener = new MenuItemListener();
181        item.addActionListener(listener);
182        return item;
183     }
184
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
185     /**
186        Creates a menu item to change the font style
187        and set its action listener.
188        @param name the name of the menu item
189        @param style the new font style
190        @return the menu item
191     */
192     public JMenuItem createStyleItem(String name, final int style)
193     {
194        JMenuItem item = new JMenuItem(name);
195        class MenuItemListener implements ActionListener
196        {
197           public void actionPerformed(ActionEvent event)
198           {
199              fontstyle = style;
200              setSampleFont();
201           }
202        }
203        ActionListener listener = new MenuItemListener();
204        item.addActionListener(listener);
205        return item;
206     }
207
```

Continued

ch17/menu/FontViewer2Frame.java (cont.)

```
208      /**
209         Sets the font of the text sample.
210      */
211      public void setSampleFont()
212      {
213         Font f = new Font(facename, fontstyle, fontsize);
214         sampleField.setFont(f);
215         sampleField.repaint();
216      }
217   }
```

Self Check 17.11

Why do `JMenu` objects not generate action events?

Answer: When you open a menu, you have not yet made a selection. Only `JMenuItem` objects correspond to selections.

Self Check 17.12

Why is the `name` parameter in the `createFaceItem` method declared as `final`?

Answer: The parameter variable is accessed in a method of an inner class.