Graphical User Interfaces

Advanced Programming

ICOM 4015

Lecture 12

Reading: Java Concepts Chapter 14

Chapter Goals

- To understand how to create frames
- To use inheritance to customize frames
- To understand how user-interface components are added to a container
- To understand the use of layout managers to arrange user-interface components in a container

Continued…

Chapter Goals

- To become familiar with common user-interface components, such as buttons, combo boxes, text areas, and menus
- To build programs that handle events from user-interface components
- To learn how to browse the Java documentation

Frame Windows

- The **`JFrame`** class

```
JFrame frame = new JFrame();
frame.setSize(300, 400);
frame.setTitle("An Empty Frame");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
```

- **`import javax.swing.*;`**

A Frame Window

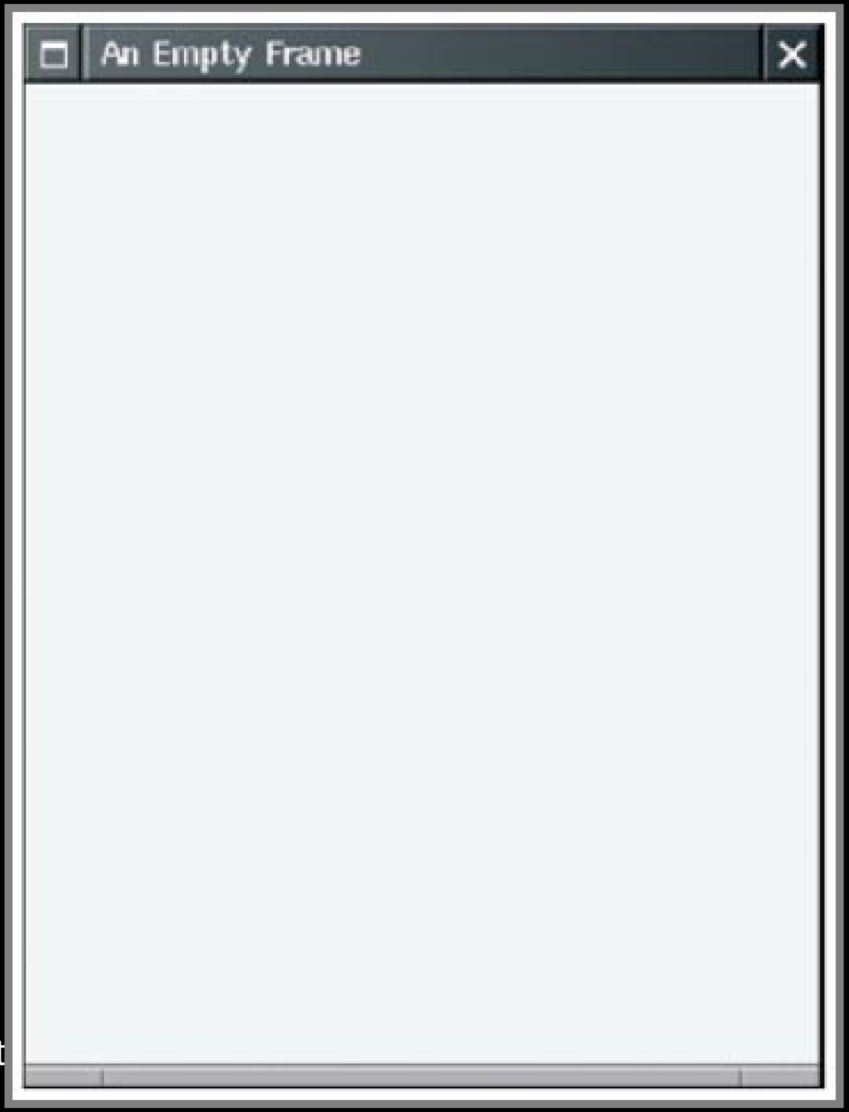


Figure 1:
A Frame Window

File EmptyFrameViewer.java

```java
import javax.swing.*;

public class EmptyFrameViewer
{
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();

      final int FRAME_WIDTH = 300;
      final int FRAME_HEIGHT = 400;

      frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      frame.setTitle("An Empty Frame");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.setVisible(true);
   }
}
```

Self Check

1. **How do you display a square frame with a title bar that reads `"Hello, World!"`?**
2. **How can a program display two frames at once?**

Answers

1. Modify the `EmptyFrameViewer` program as follows:

```
frame.setSize(300, 300);
frame.setTitle("Hello, World!");
```

2. Construct two `JFrame objects`, set each of their sizes, and call `setVisible(true)` on each of them

Basic GUI Construction

- Construct a frame
- Construct an object of your component class:

```
RectangleComponent component = new RectangleComponent();
```

- Add the component(s) to the frame

```
frame.add(component);
```

However, if you use an older version of Java (before Version 5), you must make a slightly more complicated call:

```
frame.getContentPane().add(component);
```

- Make the frame visible

Using Inheritance to Customize Frames

- Use inheritance for complex frames to make programs easier to understand
- Design a subclass of `JFrame`
- Store the components as instance fields
- Initialize them in the constructor of your subclass
- If initialization code gets complex, simply add some helper methods

Example: Investment Viewer Program

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
   This program displays the growth of an investment.
*/
public class InvestmentFrame extends JFrame
{
   public InvestmentFrame()
   {
      account = new BankAccount(INITIAL_BALANCE);

```

Continued…

Example: Investment Viewer Program

```java
        // Use instance fields for components
        resultLabel = new JLabel(
              "balance=" + account.getBalance());

        // Use helper methods
        createRateField();
        createButton();
        createPanel();

        setSize(FRAME_WIDTH, FRAME_HEIGHT);
     }

     public void createRateField()
     {
        rateLabel = new JLabel("Interest Rate: ");
        final int FIELD_WIDTH = 10;
        rateField = new JTextField(FIELD_WIDTH);
```

Continued…

Example: Investment Viewer Program

```
         rateField.setText("" + DEFAULT_RATE);
      }

      public void createButton()
      {
         button = new JButton("Add Interest");

         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());
```

Continued…

Example: Investment Viewer Program

```
            }
        }

        ActionListener listener = new AddInterestListener();
        button.addActionListener(listener);
    }

    public void createPanel()
    {
        JPanel panel = new JPanel();
        panel.add(rateLabel);
        panel.add(rateField);
        panel.add(button);
        panel.add(resultLabel);
        add(panel);
    }

```

Continued…

Example: Investment Viewer Program

```
    private JLabel rateLabel;
    private JTextField rateField;
    private JButton button;
    private JLabel resultLabel;
    private BankAccount account;

    private static final double DEFAULT_RATE = 10;
    private static final double INITIAL_BALANCE = 1000;

    private static final int FRAME_WIDTH = 500;
    private static final int FRAME_HEIGHT = 200;
}
```

Example: Investment Viewer Program

Of course, we still need a class with a `main` method:

```
import javax.swing.JFrame;

/**
   This program tests the InvestmentFrame.
*/
public class InvestmentFrameViewer
{
   public static void main(String[] args)
   {
      JFrame frame = new InvestmentFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

```

Self Check

1. How many Java source files are required by the investment viewer application when we use inheritance to define the frame class?
2. Why does the `InvestmentFrame` constructor call `setSize(FRAME_WIDTH, FRAME_HEIGHT)`, whereas the `main` method of the investment viewer class in Chapter 12 called `frame.setSize(FRAME_WIDTH, FRAME_HEIGHT)`?

Answers

1. Three: `InvestmentFrameViewer`, `InvestmentFrame`, and `BankAccount`.
2. The `InvestmentFrame` constructor adds the panel to *itself*.

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**

Continued…

Layout Management

- **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

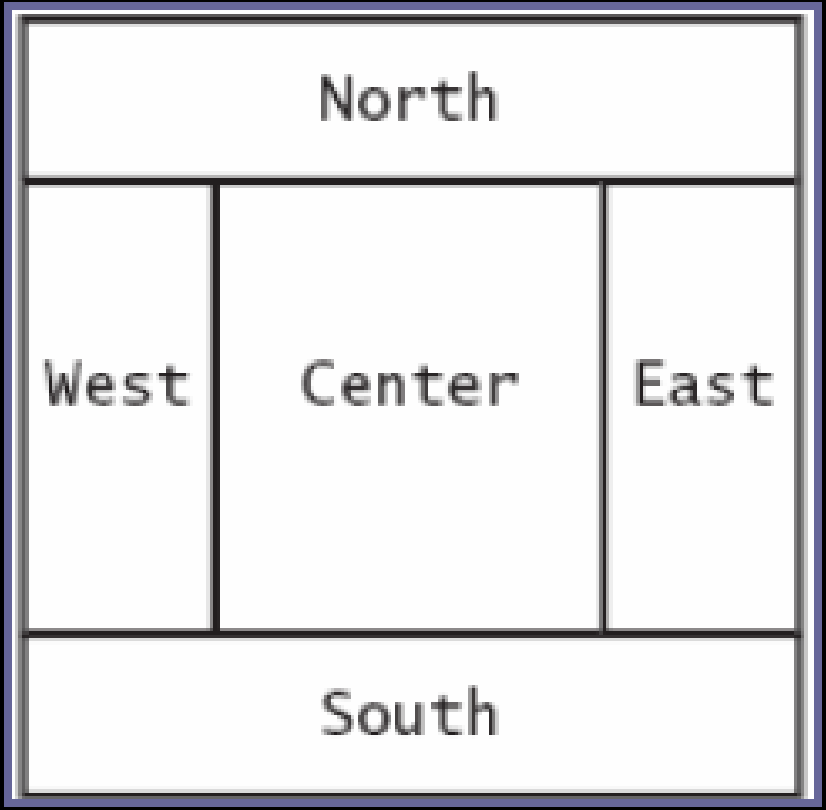


Figure 1:
Components Expand to Fill Space in the Border Layout

Continued…

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

Grid Layout

- 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

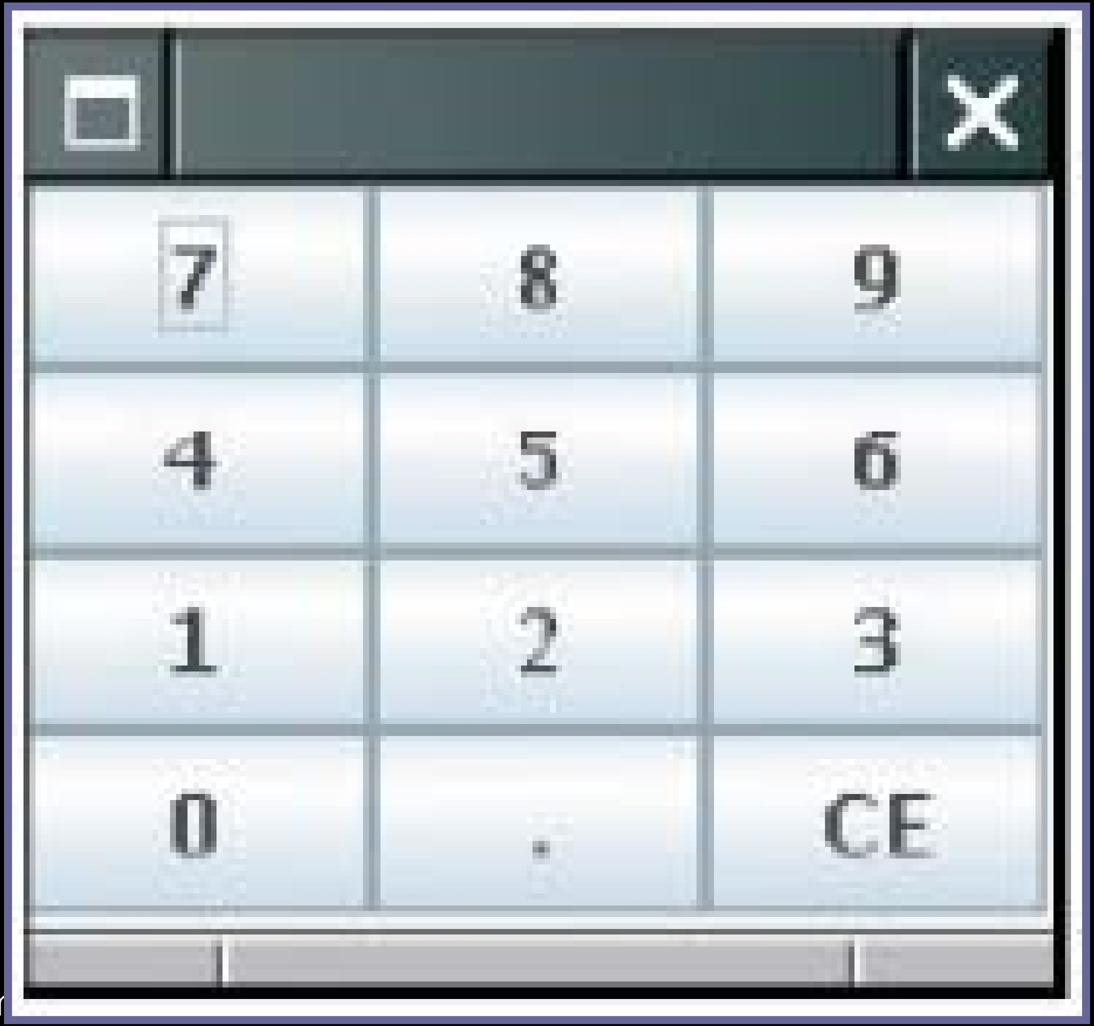


Figure 2:
The Grid Layout

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**

Continued…

Grid Bag Layout

- **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

Self Check

1. How do you add two buttons to the north area of a frame?
2. How can you stack three buttons on top of each other?

Answers

1. First add them to a panel, then add the panel to the north end of a frame.
2. Place them inside a panel with a `GridLayout` that has three rows and one column.

Choices

- Radio buttons
- Check boxes
- Combo boxes

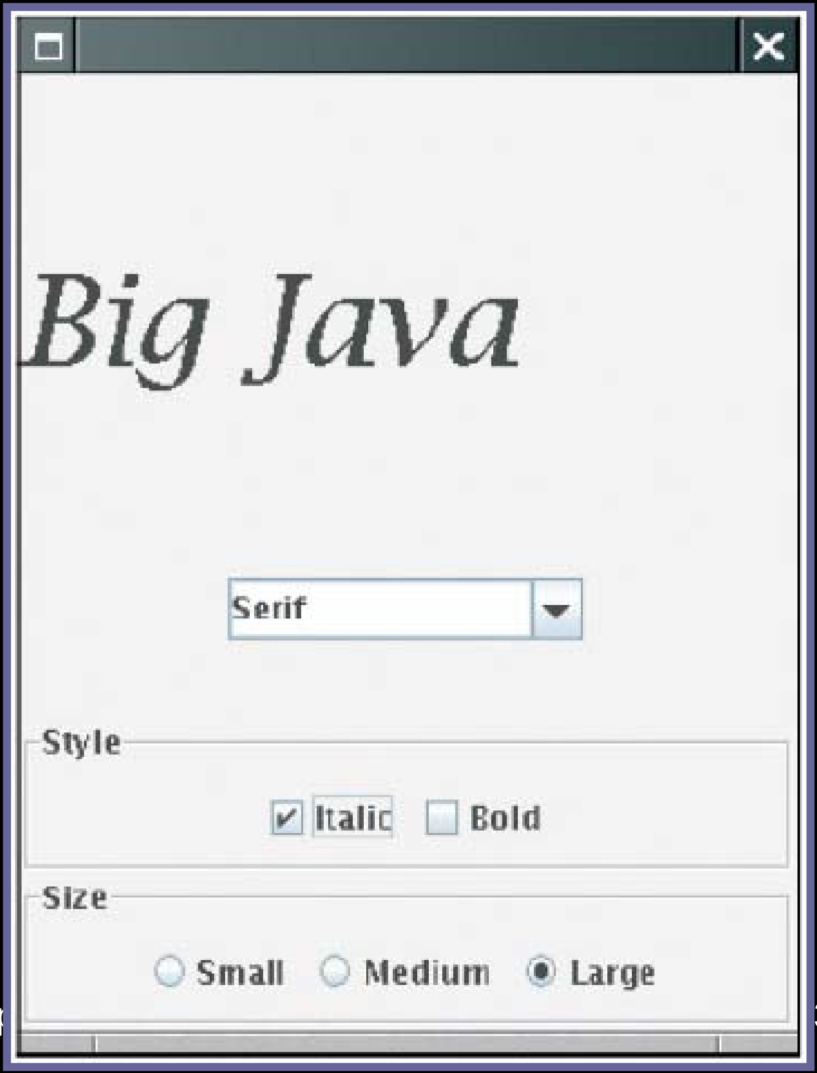


Figure 3:
A Combo Box, Check Box, and Radio Buttons

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`: theree-dimensional etched effect
- Can add a border to any component, but most commonly to panels:

```
Jpanel panel = new JPanel ();
panel.setBOrder(new EtchedBorder ());
```

Continued…

Borders

- `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

Continued…

Check Boxes

- 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

Continued…

Combo Boxes

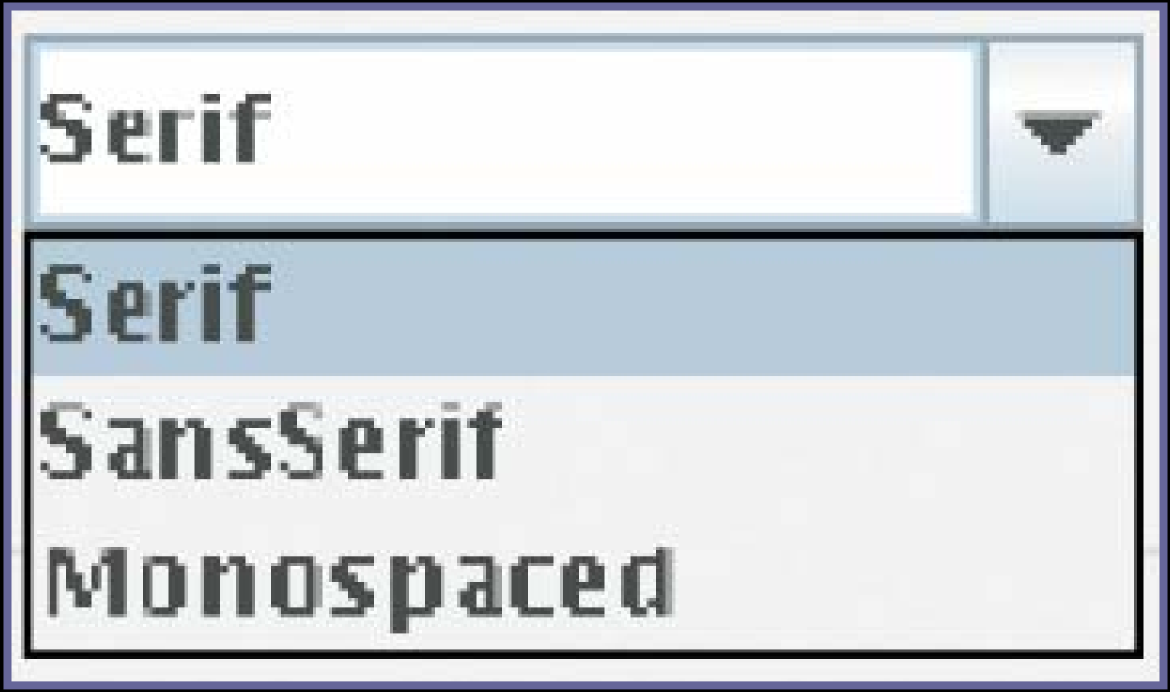


Figure 4:
An Open Combo Box

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");
. . .
```

Continued…

Combo Boxes

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

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

- Select an item with `setSelectedItem`

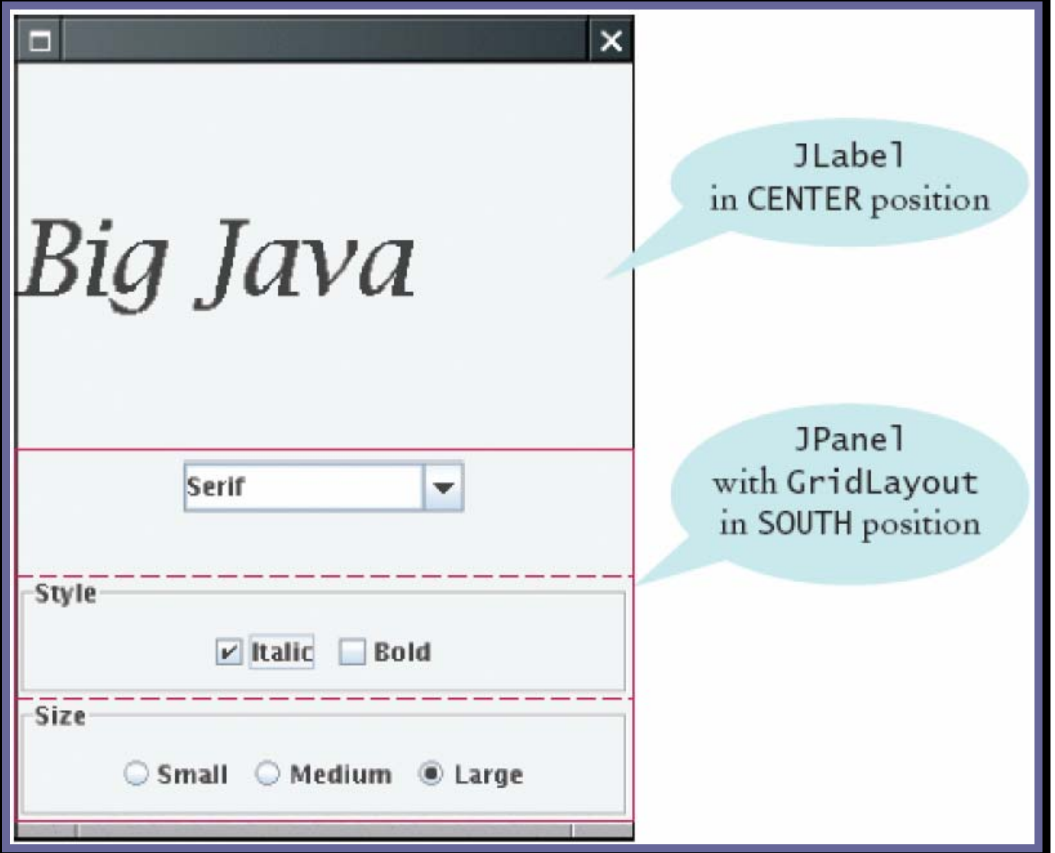
Radio Buttons, Check Boxes, and Combo Boxes

- They generate an `ActionEvent` whenever the user selects an item

Continued…

Radio Buttons, Check Boxes, and Combo Boxes

- An example: `ChoiceFrame`



Big Java
Serif
Style
Italic
Bold
Size
Small
Medium
Large
JLabel in CENTER position
JPanel with GridLayout in SOUTH position

Figure 5: The Components of the Choice Frame

Continued…

Radio Buttons, Check Boxes, and Combo Boxes

- 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 Choice Program

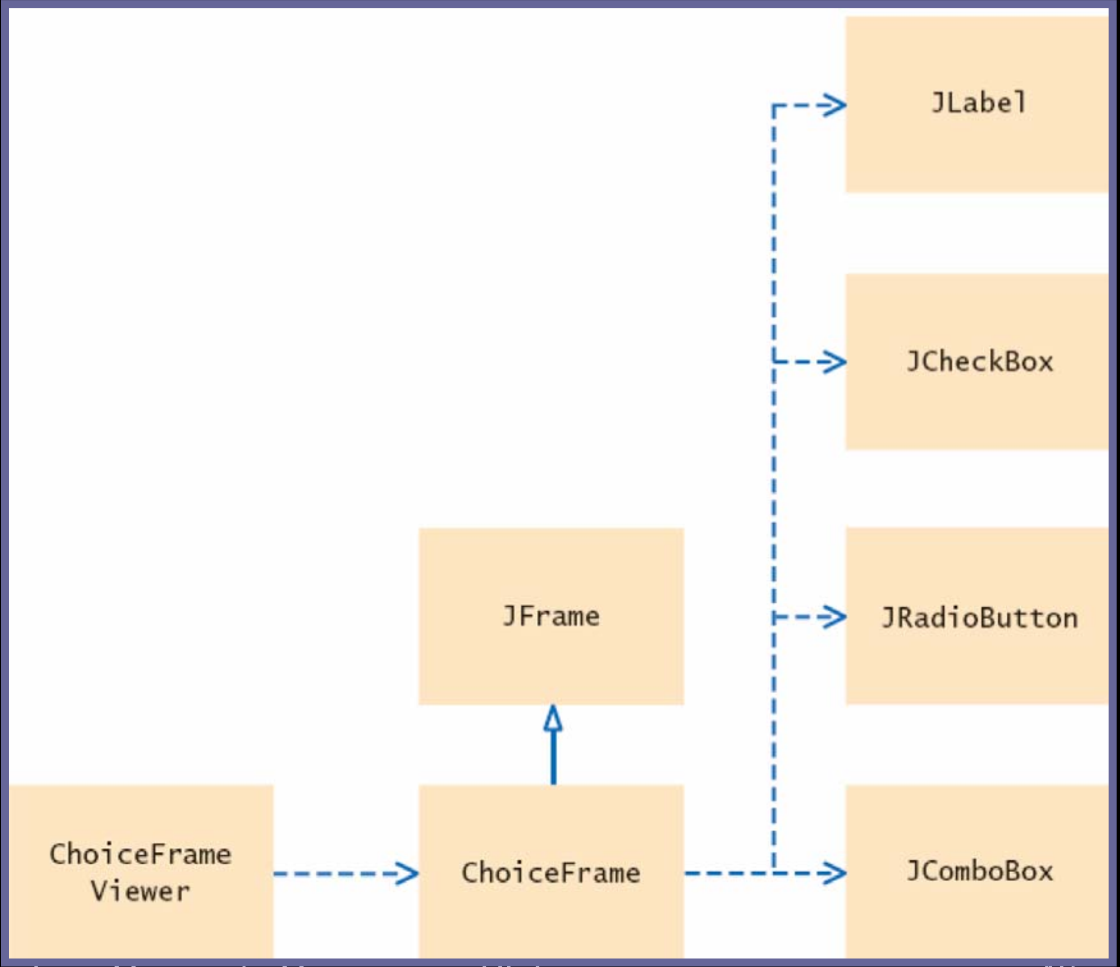


Figure 6:
Classes of the Font Choice Program

File ChoiceFrameViewer.java

```java
import javax.swing.JFrame;

/**
   This program tests the ChoiceFrame.
*/
public class ChoiceFrameViewer
{
   public static void main(String[] args)
   {
      JFrame frame = new ChoiceFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

```

File `ChoiceFrame.java`

```java
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;

```

Continued…

File ChoiceFrame.java

```java
/**
   This frame contains a text field and a control panel
   to change the font of the text.
*/
public class ChoiceFrame extends JFrame
{
   /**
      Constructs the frame.
   */
   public ChoiceFrame()
   {
      // Construct text sample
      sampleField = new JLabel("Big Java");
      add(sampleField, BorderLayout.CENTER);

```

Continued…

File `ChoiceFrame.java`

```java
      // This listener is shared among all components
      class ChoiceListener implements ActionListener
      {
         public void actionPerformed(ActionEvent event)
         {
            setSampleFont();
         }
      }

      listener = new ChoiceListener();

      createControlPanel();
      setSampleFont();
      setSize(FRAME_WIDTH, FRAME_HEIGHT);
   }

```

Continued…

File ChoiceFrame.java

```
   /**
      Creates the control panel to change the font.
   */
   public void createControlPanel()
   {
      JPanel facenamePanel = createComboBox();
      JPanel sizeGroupPanel = createCheckBoxes();
      JPanel styleGroupPanel = createRadioButtons();

      // Line up component panels

      JPanel controlPanel = new JPanel();
      controlPanel.setLayout(new GridLayout(3, 1));
      controlPanel.add(facenamePanel);
      controlPanel.add(sizeGroupPanel);
      controlPanel.add(styleGroupPanel);

```

Continued…

File ChoiceFrame.java

```java
      // Add panels to content pane

      add(controlPanel, BorderLayout.SOUTH);
   }

   /**
      Creates the combo box with the font style choices.
      @return the panel containing the combo box
   */
   public JPanel createComboBox()
   {
      facenameCombo = new JComboBox();
      facenameCombo.addItem("Serif");
      facenameCombo.addItem("SansSerif");
      facenameCombo.addItem("Monospaced");
      facenameCombo.setEditable(true);
      facenameCombo.addActionListener(listener);

```

Continued…

File ChoiceFrame.java

```java
      JPanel panel = new JPanel();
      panel.add(facenameCombo);
      return panel;
   }

   /**
      Creates the check boxes for selecting bold and
         // italic styles.
      @return the panel containing the check boxes
   */
   public JPanel createCheckBoxes()
   {
      italicCheckBox = new JCheckBox("Italic");
      italicCheckBox.addActionListener(listener);

      boldCheckBox = new JCheckBox("Bold");
      boldCheckBox.addActionListener(listener);

```

Continued…

File ChoiceFrame.java

```
      JPanel panel = new JPanel();
      panel.add(italicCheckBox);
      panel.add(boldCheckBox);
      panel.setBorder
         (new TitledBorder(new EtchedBorder(), "Style"));

      return panel;
   }

   /**
      Creates the radio buttons to select the font size
      @return the panel containing the radio buttons
   */
   public JPanel createRadioButtons()
   {
      smallButton = new JRadioButton("Small");
      smallButton.addActionListener(listener);
```

Continued…

File ChoiceFrame.java

```

      mediumButton = new JRadioButton("Medium");
      mediumButton.addActionListener(listener);

      largeButton = new JRadioButton("Large");
      largeButton.addActionListener(listener);
      largeButton.setSelected(true);

      // Add radio buttons to button group

      ButtonGroup group = new ButtonGroup();
      group.add(smallButton);
      group.add(mediumButton);
      group.add(largeButton);

```

Continued…

File ChoiceFrame.java

```java
      JPanel panel = new JPanel();
      panel.add(smallButton);
      panel.add(mediumButton);
      panel.add(largeButton);
      panel.setBorder
            (new TitledBorder(new EtchedBorder(), "Size"));

      return panel;
   }

   /**
      Gets user choice for font name, style, and size
      and sets the font of the text sample.
   */
   public void setSampleFont()
   {
```

Continued…

File ChoiceFrame.java

```java
      // Get font name
      String facename
            = (String) facenameCombo.getSelectedItem();

      // Get font style

      int style = 0;
      if (italicCheckBox.isSelected())
         style = style + Font.ITALIC;
      if (boldCheckBox.isSelected())
         style = style + Font.BOLD;

      // Get font size

      int size = 0;

```

Continued…

File `ChoiceFrame.java`

```java
         final int SMALL_SIZE = 24;
         final int MEDIUM_SIZE = 36;
         final int LARGE_SIZE = 48;

         if (smallButton.isSelected())
            size = SMALL_SIZE;
         else if (mediumButton.isSelected())
            size = MEDIUM_SIZE;
         else if (largeButton.isSelected())
            size = LARGE_SIZE;

         // Set font of text field

         sampleField.setFont(new Font(facename, style, size));
         sampleField.repaint();
      }
```

Continued…

File ChoiceFrame.java

```

    private JLabel sampleField;
    private JCheckBox italicCheckBox;
    private JCheckBox boldCheckBox;
    private JRadioButton smallButton;
    private JRadioButton mediumButton;
    private JRadioButton largeButton;
    private JComboBox facenameCombo;
    private ActionListener listener;

    private static final int FRAME_WIDTH = 300;
    private static final int FRAME_HEIGHT = 400;
}
```

Continued…

Self Check

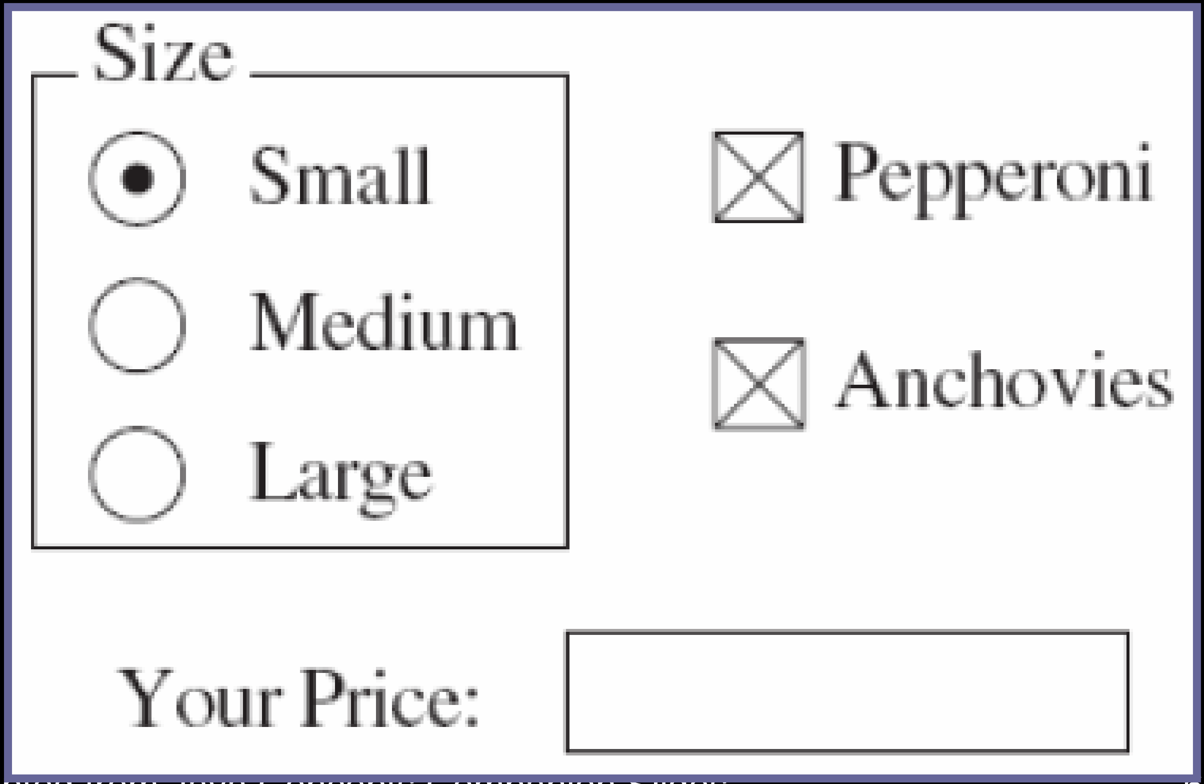
1. **What is the advantage of a `JComboBox` over a set of radio buttons? What is the disadvantage?**
2. **Why do all user interface components in the `ChoiceFrame` class share the same listener?**
3. **Why was the combo box placed inside a panel? What would have happened if it had been added directly to the control panel?**

Answers

1. 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.
2. When any of the component settings is changed, the program simply queries all of them and updates the label.
3. To keep it from growing too large. It would have grown to the same width and height as the two panels below it

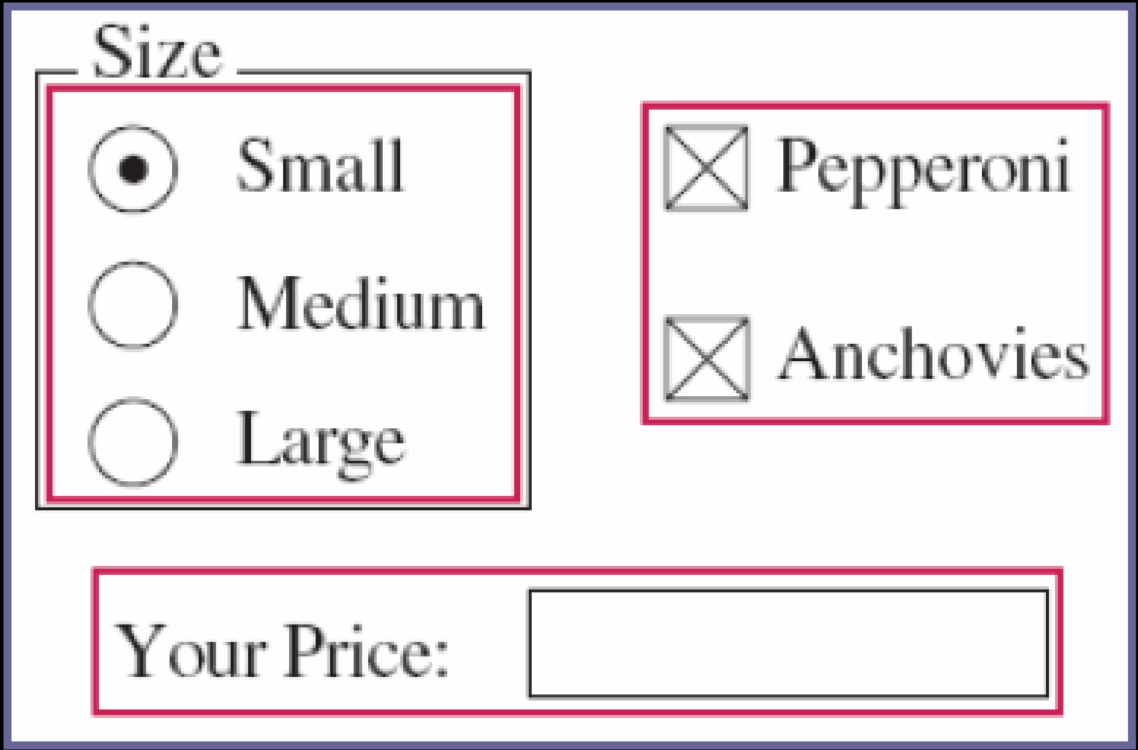
Advanced Topic: Layout Management

- Step 1: Make a sketch of your desired component layout



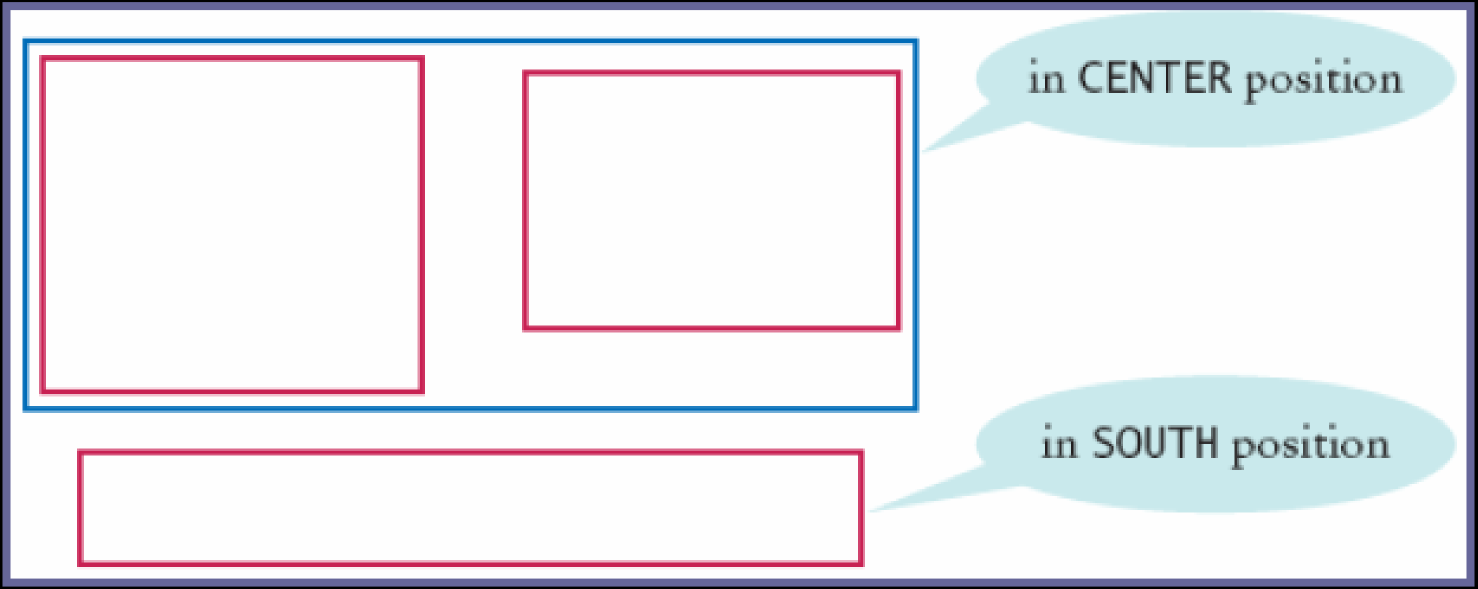
Advanced Topic: Layout Management

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



Advanced Topic: Layout Management

- Step 3: Identify layouts for each group
- Step 4: Group the groups together



- Step 5: Write the code to generate the layout

Menus

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

Menus

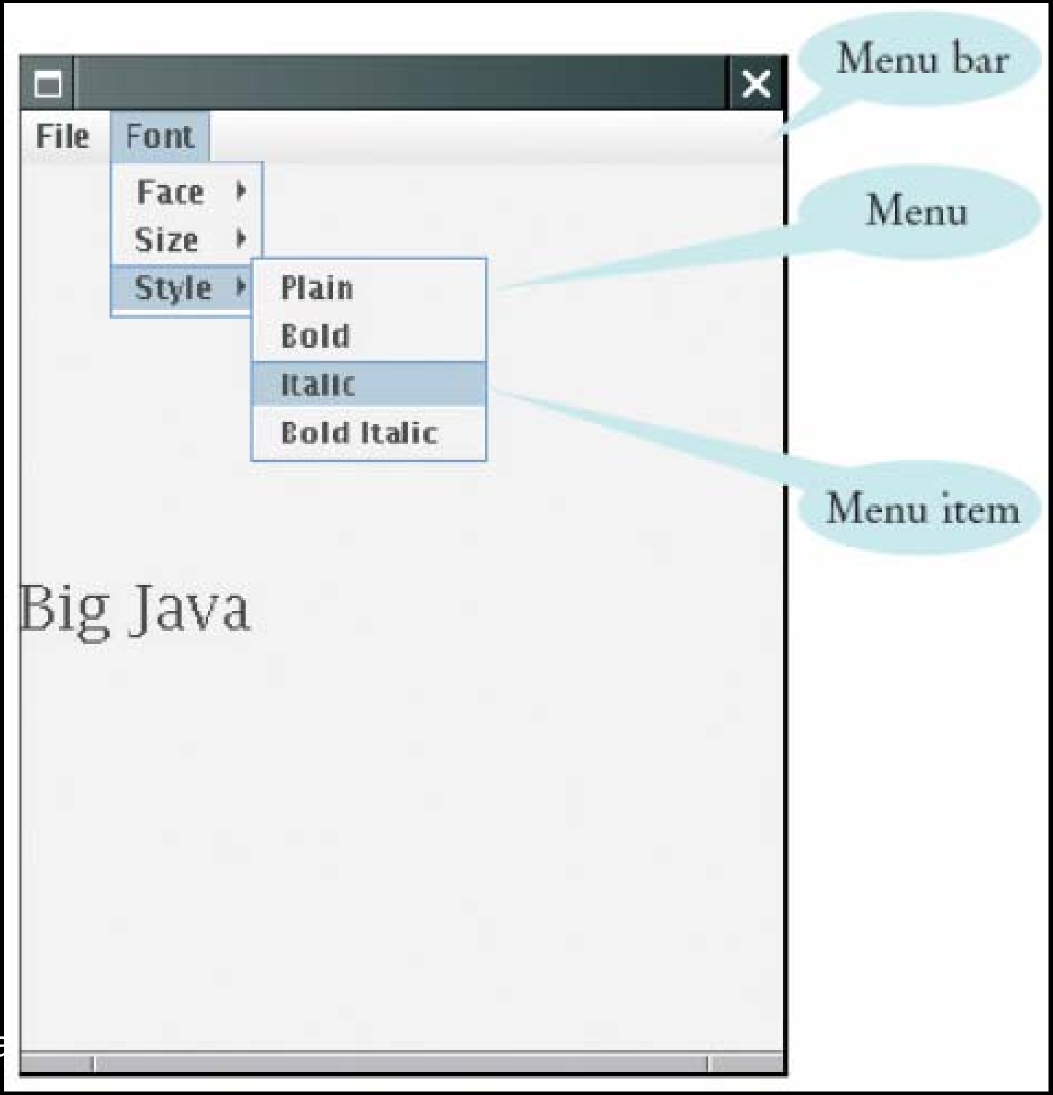


Figure 7:
Pull-Down Menus

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

Continued…

Menu Items

- 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`

File `MenuFrameViewer.java`

```java
import javax.swing.JFrame;

/**
   This program tests the MenuFrame.
*/
public class MenuFrameViewer
{
   public static void main(String[] args)
   {
      JFrame frame = new MenuFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

```

File `MenuFrame.java`

```
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
```

Continued…

File MenuFrame.java

```java
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;

/**
   This frame has a menu with commands to change the font
   of a text sample.
*/
public class MenuFrame extends JFrame
{
   /**
      Constructs the frame.
   */
   public MenuFrame()
   {
      // Construct text sample
      sampleField = new JLabel("Big Java");
      add(sampleField, BorderLayout.CENTER);

```

Continued…

File MenuFrame.java

```
        // Construct menu
        JMenuBar menuBar = new JMenuBar();
        setJMenuBar(menuBar);
        menuBar.add(createFileMenu());
        menuBar.add(createFontMenu());

        facename = "Serif";
        fontsize = 24;
        fontstyle = Font.PLAIN;

        setSampleFont();
        setSize(FRAME_WIDTH, FRAME_HEIGHT);
    }

    /**
        Creates the File menu.
        @return the menu
    */
```

Continued…

File MenuFrame.java

```java
   public JMenu createFileMenu()
   {
      JMenu menu = new JMenu("File");
      menu.add(createFileExitItem());
      return menu;
   }

   /**
      Creates the File->Exit menu item and sets its
         // action listener.
      @return the menu item
   */
   public JMenuItem createFileExitItem()
   {
      JMenuItem item = new JMenuItem("Exit");
      class MenuItemListener implements ActionListener
      {
         public void actionPerformed(ActionEvent event)
```

Continued…

File MenuFrame.java

```
            {
               System.exit(0);
            }
         }
         ActionListener listener = new MenuItemListener();
         item.addActionListener(listener);
         return item;
      }

      /**
         Creates the Font submenu.
         @return the menu
      */
      public JMenu createFontMenu()
      {
         JMenu menu = new JMenu("Font");
         menu.add(createFaceMenu());
```

Continued…

File MenuFrame.java

```java
        menu.add(createSizeMenu());
        menu.add(createStyleMenu());
        return menu;
    }

    /**
        Creates the Face submenu.
        @return the menu
    */
    public JMenu createFaceMenu()
    {
        JMenu menu = new JMenu("Face");
        menu.add(createFaceItem("Serif"));
        menu.add(createFaceItem("SansSerif"));
        menu.add(createFaceItem("Monospaced"));
        return menu;
    }

```

Continued…

File MenuFrame.java

```
   /**
      Creates the Size submenu.
      @return the menu
   */
   public JMenu createSizeMenu()
   {
      JMenu menu = new JMenu("Size");
      menu.add(createSizeItem("Smaller", -1));
      menu.add(createSizeItem("Larger", 1));
      return menu;
   }

   /**
      Creates the Style submenu.
      @return the menu
   */
   public JMenu createStyleMenu()
   {
```

Continued…

File MenuFrame.java

```
      JMenu menu = new JMenu("Style");
      menu.add(createStyleItem("Plain", Font.PLAIN));
      menu.add(createStyleItem("Bold", Font.BOLD));
      menu.add(createStyleItem("Italic", Font.ITALIC));
      menu.add(createStyleItem("Bold Italic", Font.BOLD
            + Font.ITALIC));
      return menu;
   }


   /**
      Creates a menu item to change the font face and
         // set its action listener.
      @param name the name of the font face
      @return the menu item
   */
   public JMenuItem createFaceItem(final String name)
   {
```

Continued…

File `MenuFrame.java`

```java
      JMenuItem item = new JMenuItem(name);
      class MenuItemListener implements ActionListener
      {
         public void actionPerformed(ActionEvent event)
         {
            facename = name;
            setSampleFont();
         }
      }
      ActionListener listener = new MenuItemListener();
      item.addActionListener(listener);
      return item;
   }

```

Continued…

File MenuFrame.java

```
   /**
      Creates a menu item to change the font size
      and set its action listener.
      @param name the name of the menu item
      @param ds the amount by which to change the size
      @return the menu item
   */
   public JMenuItem createSizeItem(String name, final int ds)
   {
      JMenuItem item = new JMenuItem(name);
      class MenuItemListener implements ActionListener
      {
         public void actionPerformed(ActionEvent event)
         {
            fontsize = fontsize + ds;
            setSampleFont();
         }
      }
```

Continued…

File MenuFrame.java

```
         ActionListener listener = new MenuItemListener();
         item.addActionListener(listener);
         return item;
      }

      /**
         Creates a menu item to change the font style
         and set its action listener.
         @param name the name of the menu item
         @param style the new font style
         @return the menu item
      */
      public JMenuItem createStyleItem(String name,
         final int style)
      {
         JMenuItem item = new JMenuItem(name);
         class MenuItemListener implements ActionListener
         {
```

Continued…

File `MenuFrame.java`

```java
         public void actionPerformed(ActionEvent event)
         {
            fontstyle = style;
            setSampleFont();
         }
      }
      ActionListener listener = new MenuItemListener();
      item.addActionListener(listener);
      return item;
   }

   /**
      Sets the font of the text sample.
   */
   public void setSampleFont()
   {
```

Continued…

File `MenuFrame.java`

```
            Font f = new Font(facename, fontstyle, fontsize);
            sampleField.setFont(f);
            sampleField.repaint();
        }

    private JLabel sampleField;
    private String facename;
    private int fontstyle;
    private int fontsize;

    private static final int FRAME_WIDTH = 300;
    private static final int FRAME_HEIGHT = 400;
}


```

Self Check

1. Why do `JMenu` objects not generate action events?
2. Why is the name parameter in the `createFaceItem` method declared as `final`?

Answers

1. When you open a menu, you have not yet made a selection. Only `JMenuItem` objects correspond to selections.
2. The parameter variable is accessed in a method of an inner class.

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

Continued…

Text Areas

- 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
```

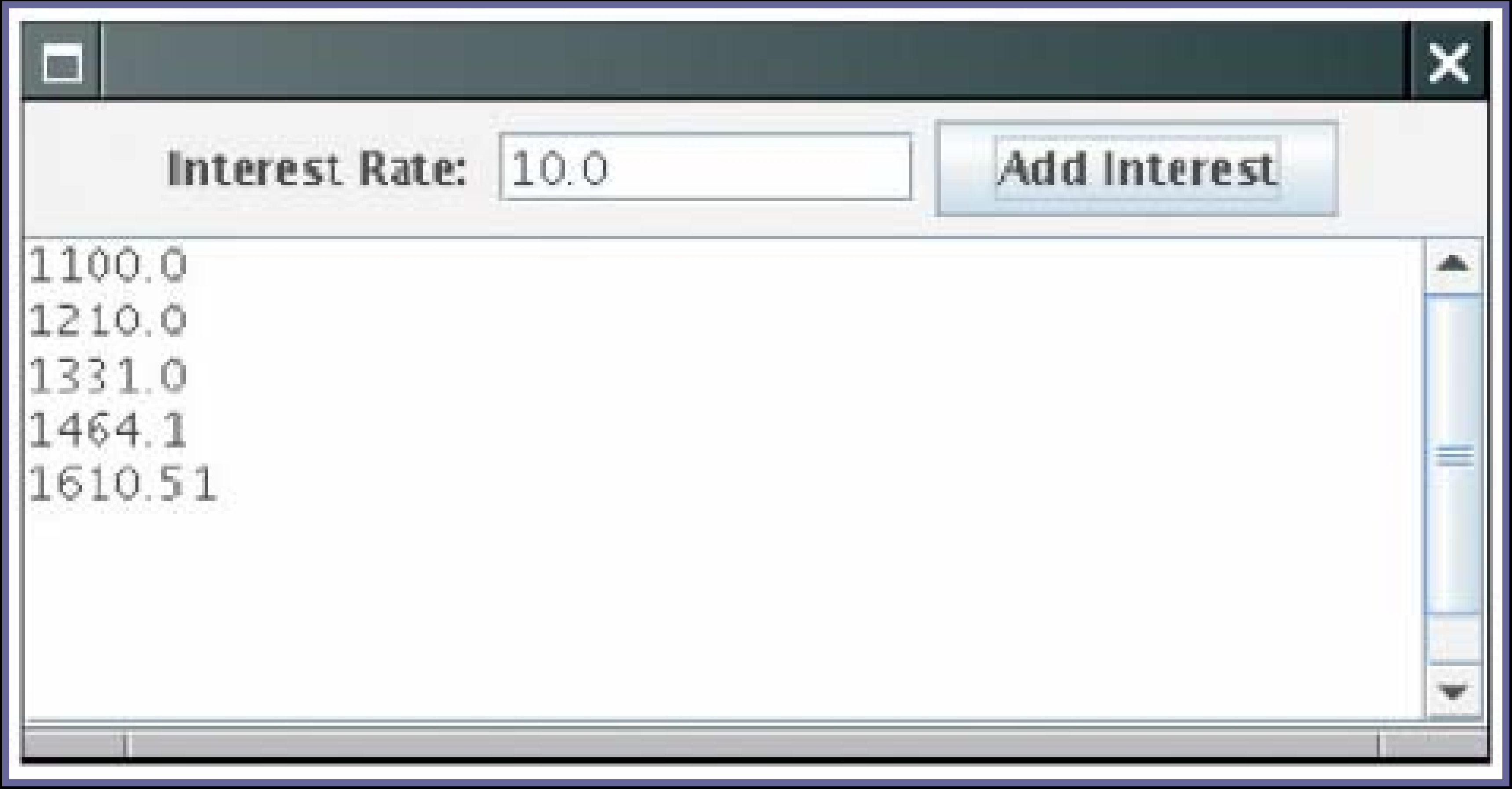
Text Areas

To add scroll bars to a text area:

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

Continued…

Text Areas



```
10.0
```

```
1100.0
1210.0
1331.0
1464.1
1610.51
```

Figure 8:
The `TextAreaViewer` Application

File TextAreaViewer.java

```java
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
   This program shows a frame with a text area that
   displays the growth of an investment.
*/
public class TextAreaViewer
{
```

Continued…

File TextAreaViewer.java

```
   public static void main(String[] args)
   {
      JFrame frame = new JFrame();

      // The application adds interest to this bank account
      final BankAccount account =
         new BankAccount(INITIAL_BALANCE);
      // The text area for displaying the results
      final int AREA_ROWS = 10;
      final int AREA_COLUMNS = 30;

      final JTextArea textArea = new JTextArea(
            AREA_ROWS, AREA_COLUMNS);
      textArea.setEditable(false);
      JScrollPane scrollPane = new JScrollPane(textArea);

      // The label and text field for entering the
         // interest rate
```

Continued…

File TextAreaViewer.java

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

      final int FIELD_WIDTH = 10;
      final JTextField rateField =
         new JTextField(FIELD_WIDTH);
      rateField.setText("" + DEFAULT_RATE);

      // The button to trigger the calculation
      JButton calculateButton = new JButton("Add Interest");

      // The panel that holds the input components
      JPanel northPanel = new JPanel();
      northPanel.add(rateLabel);
      northPanel.add(rateField);
      northPanel.add(calculateButton);

      frame.add(northPanel, BorderLayout.NORTH);
      frame.add(scrollPane);

```

Continued…

File TextAreaViewer.java

```java
      class CalculateListener implements ActionListener
      {
         public void actionPerformed(ActionEvent event)
         {
            double rate = Double.parseDouble(
                  rateField.getText());
            double interest = account.getBalance()
                  * rate / 100;
            account.deposit(interest);
            textArea.append(account.getBalance() + "\n");
         }
      }

      ActionListener listener = new CalculateListener();
      calculateButton.addActionListener(listener);

```

Continued…

File TextAreaViewer.java

```java
        frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    private static final double DEFAULT_RATE = 10;
    private static final double INITIAL_BALANCE = 1000;

    private static final int FRAME_WIDTH = 400;
    private static final int FRAME_HEIGHT = 200;
}
```

Self Check

1. **What is the difference between a text field and a text area?**
2. **Why did the `TextAreaViewer` program call `textArea.setEditable(false)`?**
3. **How would you modify the `TextAreaViewer` program if you didn't want to use scroll bars?**

Answers

1. A text field holds a single line of text; a text area holds multiple lines.
2. The text area is intended to display the program output. It does not collect user input.
3. Don't construct a `JScrollPane` and add the `textArea` object directly to the frame.

Exploring the Swing Documentation

- For more sophisticated effects, explore the Swing documentation
- The documentation can be quite intimidating at first glance
- Next example will show how to use the documentation to your advantage

Example: A Color Mixer

- It should be fun to mix your own colors, with a slider for the red, green, and blue values

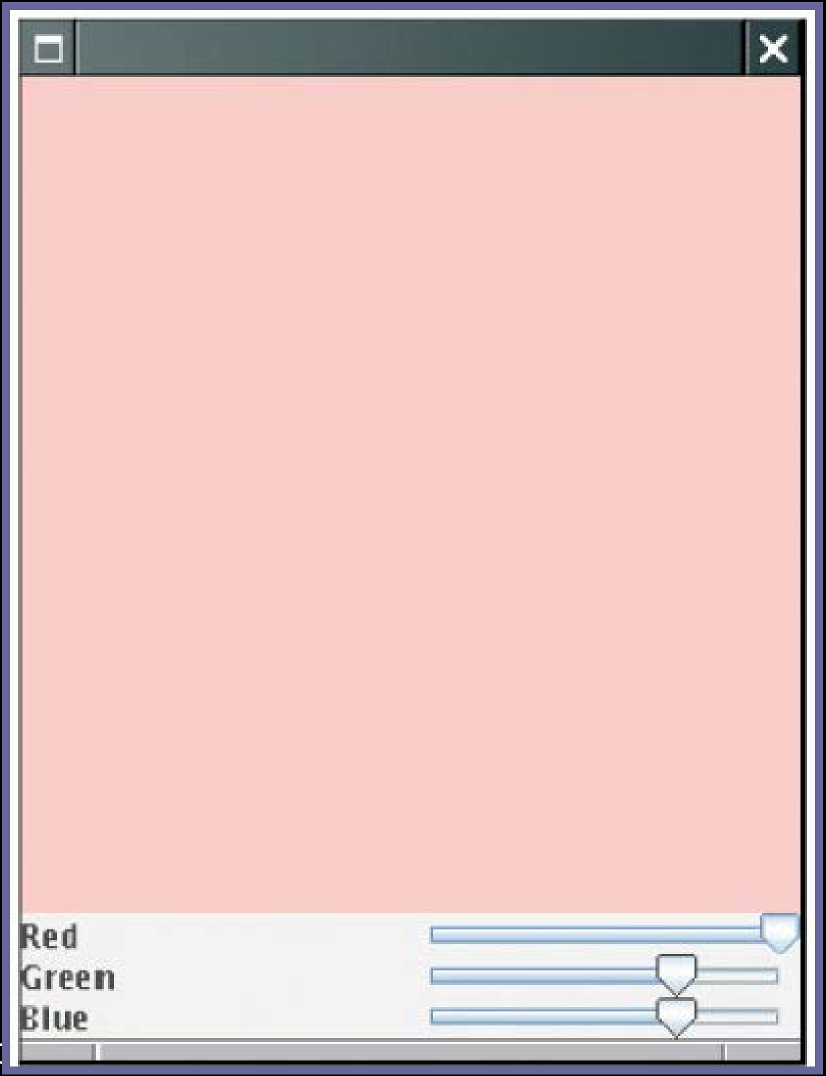


Figure 9:
A Color Mixer

Example: A Color Mixer

- **How do you know if there is a slider?**
  - Buy a book that illustrates all Swing components
  - Run sample application included in the JDK that shows off all Swing components
  - Look at the names of all of the classes that start with J
    - `JSlider` seems like a good candidate

Continued…

Example: A Color Mixer

- **Next, ask a few questions:**
    - How do I construct a `JSlider`?
    - How can I get notified when the user has moved it?
    - How can I tell to which value the user has set it?
- **After mastering sliders, you can find out how to set tick marks, etc.**

Continued…

The Swing Demo Set

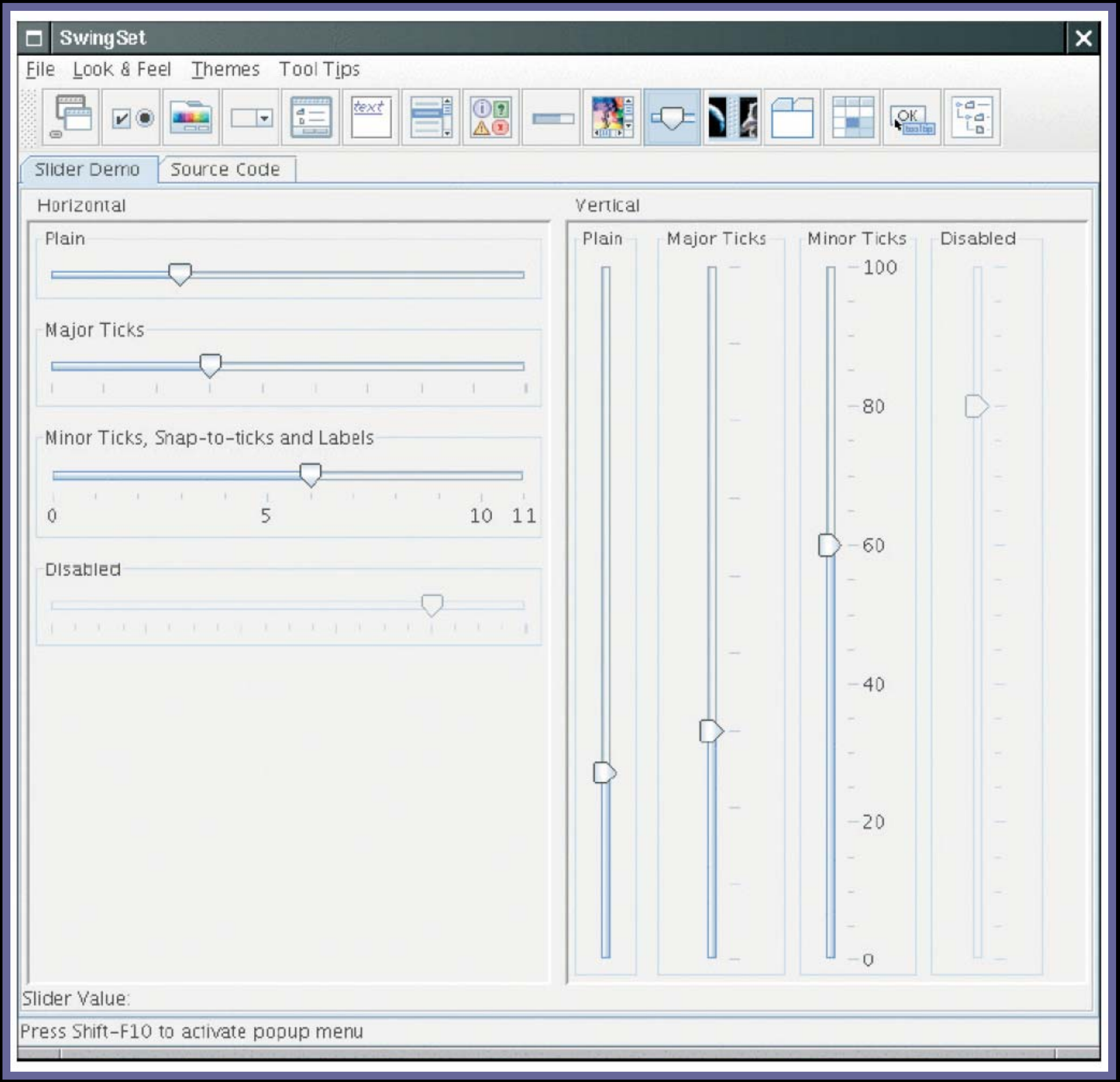


Figure 9:
The SwingSet Demo

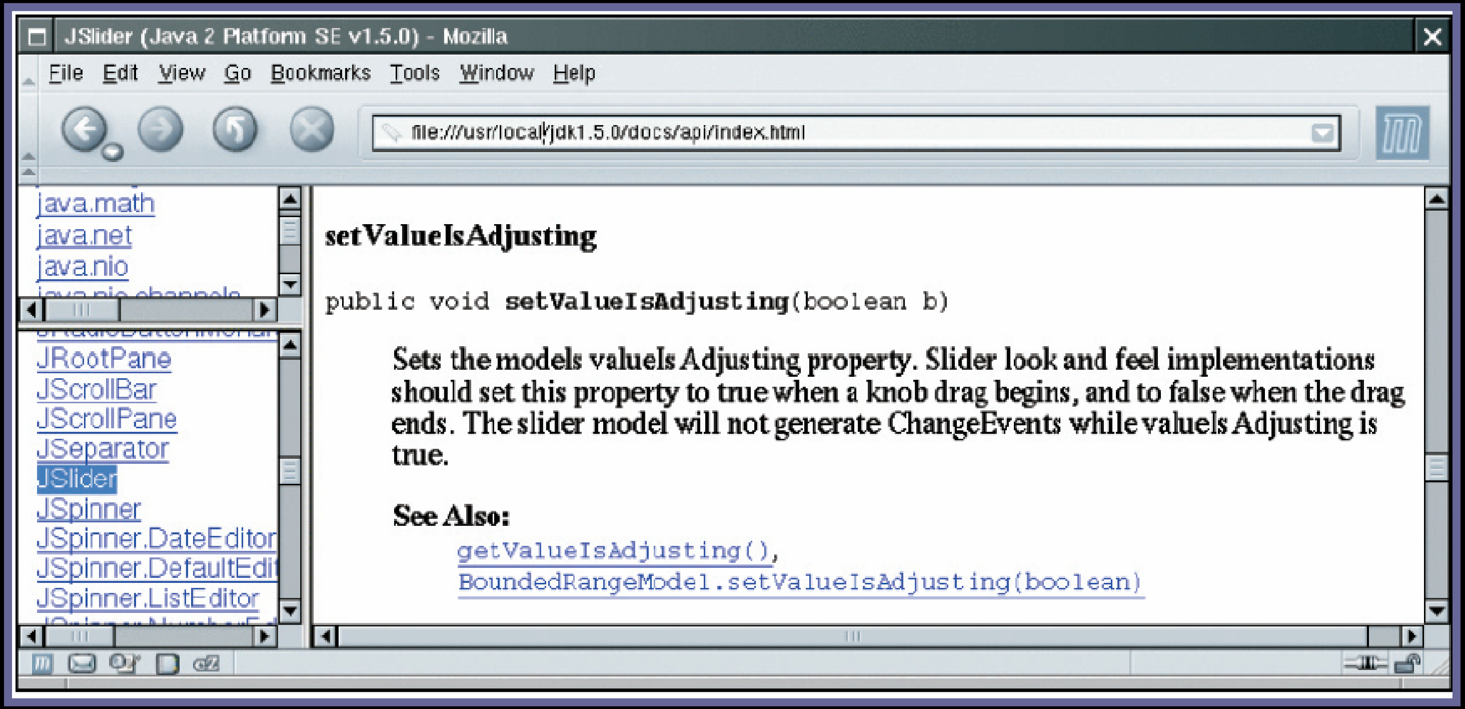
Adapted from

Example: A Color Mixer

- There are over 50 methods in `JSlider` class and over 250 inherited methods
- Some method descriptions look scary

Continued…

Example: A Color Mixer



setValueIsAdjusting

```
public void setValueIsAdjusting(boolean b)
```

Sets the models valueIsAdjusting property. Slider look and feel implementations should set this property to true when a knob drag begins, and to false when the drag ends. The slider model will not generate ChangeEvents while valueIsAdjusting is true.

See Also:
`getValueIsAdjusting()`, `BoundedRangeModel.setValueIsAdjusting(boolean)`

Figure 11: A Mysterious Method Description from the API Documentation

- **Develop the ability to separate fundamental concepts from ephemeral minutiae**

How do I construct a `JSlider`?

- Look at the Java version 5.0 API documentation
- There are six constructors for the `JSlider` class
- Learn about one or two of them
- Strike a balance somewhere between the trivial and the bizarre

Continued…

How do I construct a `JSlider`?

- **Too limited:**

```
public JSlider()
```

Creates a horizontal slider with the range 0 to 100 and an initial value of 50

- **Bizarre:**

```
public JSlider(BoundedRangeModel brm)
```

Creates a horizontal slider using the specified `BoundedRangeModel`

Continued…

How do I construct a `JSlider`?

- **Useful:**

```
public JSlider(int min, int max, int value)
```

Creates a horizontal slider using the specified min, max, and value.

How Can I Get Notified When the User Has Moved a `JSlider`?

- There is no `addActionListener` method
- There is a method

```
public void addChangeListener(ChangeListener l)
```

- Click on the `ChangeListener` link to learn more
- It has a single method:

```
void stateChanged(ChangeEvent e)
```

Continued…

How Can I Get Notified When the User Has Moved a `JSlider`?

- **Apparently, method is called whenever user moves the slider**
- **What is a `ChangeEvent`?**
  - It inherits `getSource` method from superclass `EventObject`
  - `getSource`: tells us which component generated this event

How Can I Tell to Which Value the User Has Set a `JSlider`?

- **Now we have a plan:**
  - Add a change event listener to each slider
  - When slider is changed, `stateChanged` method is called
  - Find out the new value of the slider
  - Recompute color value
  - Repaint color panel

Continued…

How Can I Tell to Which Value the User Has Set a `JSlider`?

- Need to get the current value of the slider
- Look at all the methods that start with get; you find:

```
public int getValue()
```

Returns the slider's value.

The Components of the `SliderFrame`

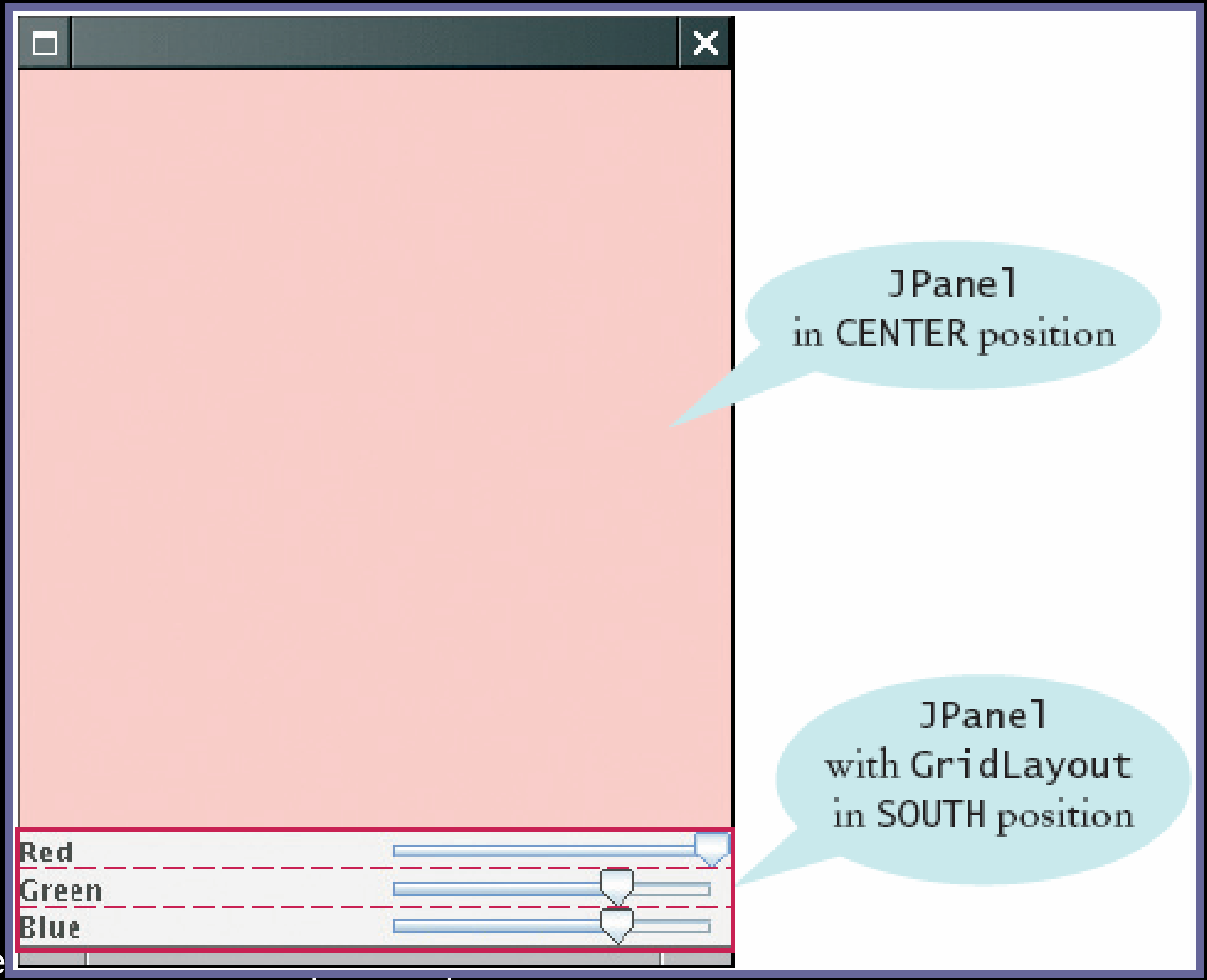

JPanel in CENTER position
JPanel with GridLayout in SOUTH position
Red
Green
Blue

Figure 12:
The Components of the `SliderFrame`

Classes of the `SliderFrameViewer` Program

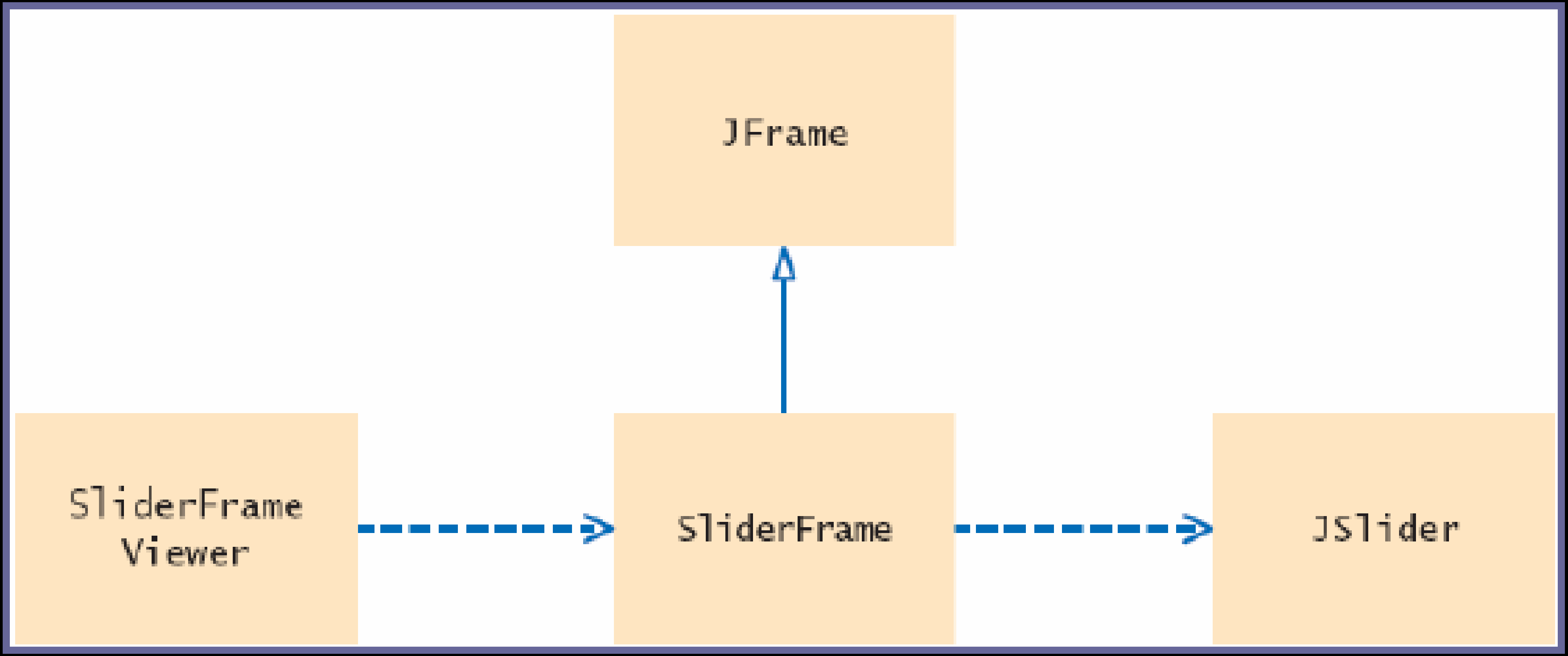


Figure 13:
Classes of the `SliderFrameViewer` Program

File `SliderFrameViewer.java`

```java
import javax.swing.JFrame;

public class SliderFrameViewer
{
   public static void main(String[] args)
   {
      SliderFrame frame = new SliderFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

```

File `SliderFrame.java`

```java
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

public class SliderFrame extends JFrame
{
   public SliderFrame()
   {
      colorPanel = new JPanel();

```

Continued…

File SliderFrame.java

```java
        add(colorPanel, BorderLayout.CENTER);
        createControlPanel();
        setSampleColor();
        setSize(FRAME_WIDTH, FRAME_HEIGHT);
    }

    public void createControlPanel()
    {
        class ColorListener implements ChangeListener
        {
            public void stateChanged(ChangeEvent event)
            {
                setSampleColor();
            }
        }

```

Continued…

File SliderFrame.java

```
      ChangeListener listener = new ColorListener();

      redSlider = new JSlider(0, 100, 100);
      redSlider.addChangeListener(listener);

      greenSlider = new JSlider(0, 100, 70);
      greenSlider.addChangeListener(listener);

      blueSlider = new JSlider(0, 100, 70);
      blueSlider.addChangeListener(listener);

      JPanel controlPanel = new JPanel();
      controlPanel.setLayout(new GridLayout(3, 2));

      controlPanel.add(new JLabel("Red"));
      controlPanel.add(redSlider);

```

Continued…

File SliderFrame.java

```
      controlPanel.add(new JLabel("Green"));
      controlPanel.add(greenSlider);

      controlPanel.add(new JLabel("Blue"));
      controlPanel.add(blueSlider);

      add(controlPanel, BorderLayout.SOUTH);
   }

   /**
      Reads the slider values and sets the panel to
      the selected color.
   */
   public void setSampleColor()
   {
      // Read slider values

```

Continued…

File SliderFrame.java

```
            float red = 0.01F * redSlider.getValue();
            float green = 0.01F * greenSlider.getValue();
            float blue = 0.01F * blueSlider.getValue();

            // Set panel background to selected color

            colorPanel.setBackground(new Color(red, green, blue));
            colorPanel.repaint();
        }

        private JPanel colorPanel;
        private JSlider redSlider;
        private JSlider greenSlider;
        private JSlider blueSlider;

        private static final int FRAME_WIDTH = 300;
        private static final int FRAME_HEIGHT = 400;
    }
```

Continued…

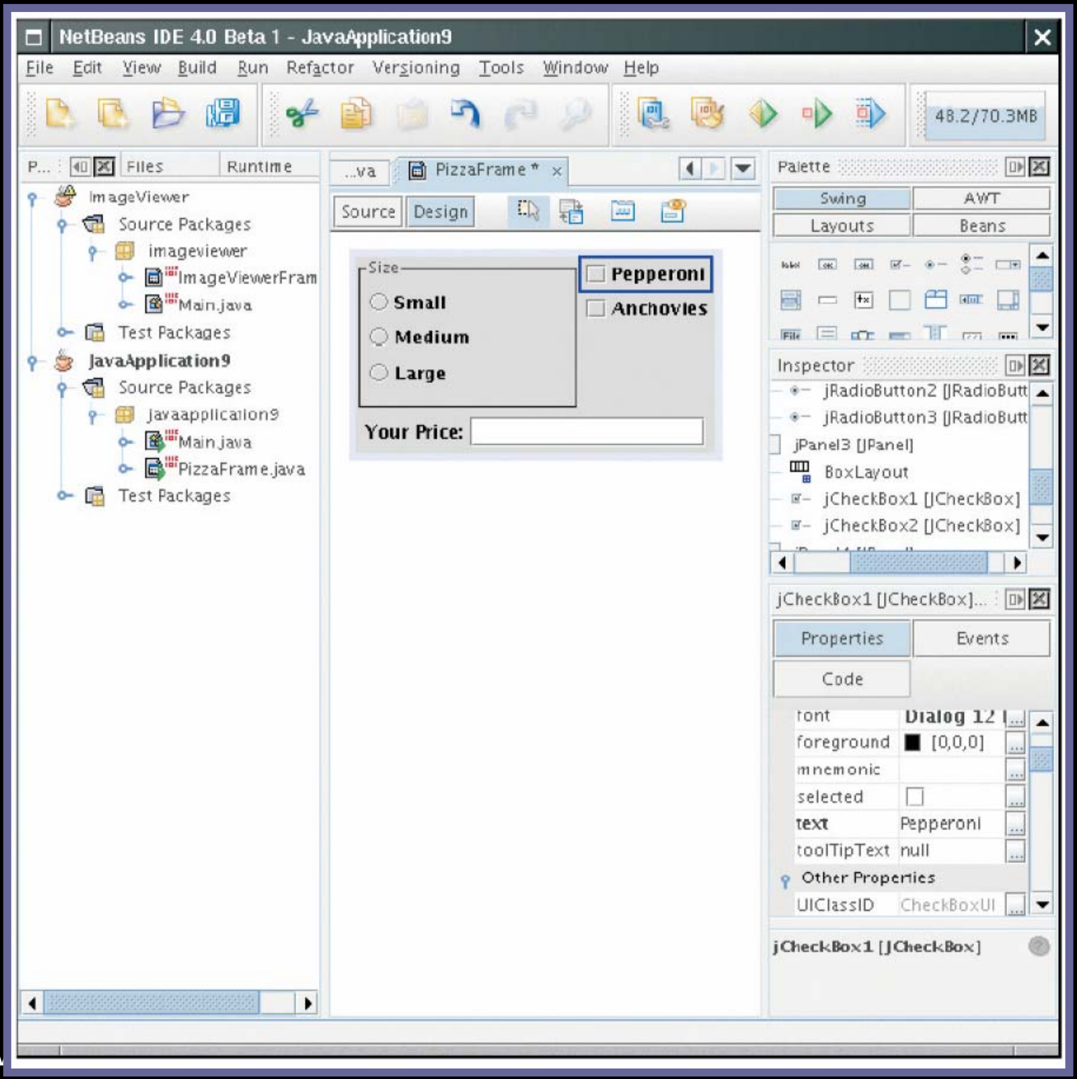
Self Check

1. **Suppose you want to allow users to pick a color from a color dialog box. Which class would you use? Look in the API documentation.**
2. **Why does a slider emit change events and not action events?**

Answers

- `JColorChooser`.
- Action events describe one-time changes, such as button clicks. Change events describe continuous changes.

Visual Programming



font	Dialog 12
foreground	■ [0,0,0]
mnemonic	
selected	☐
text	Pepperoni
toolTipText	null
UIClassID	CheckBoxUI

Figure 14:
A Visual Programming Environment