Introduction to GUI

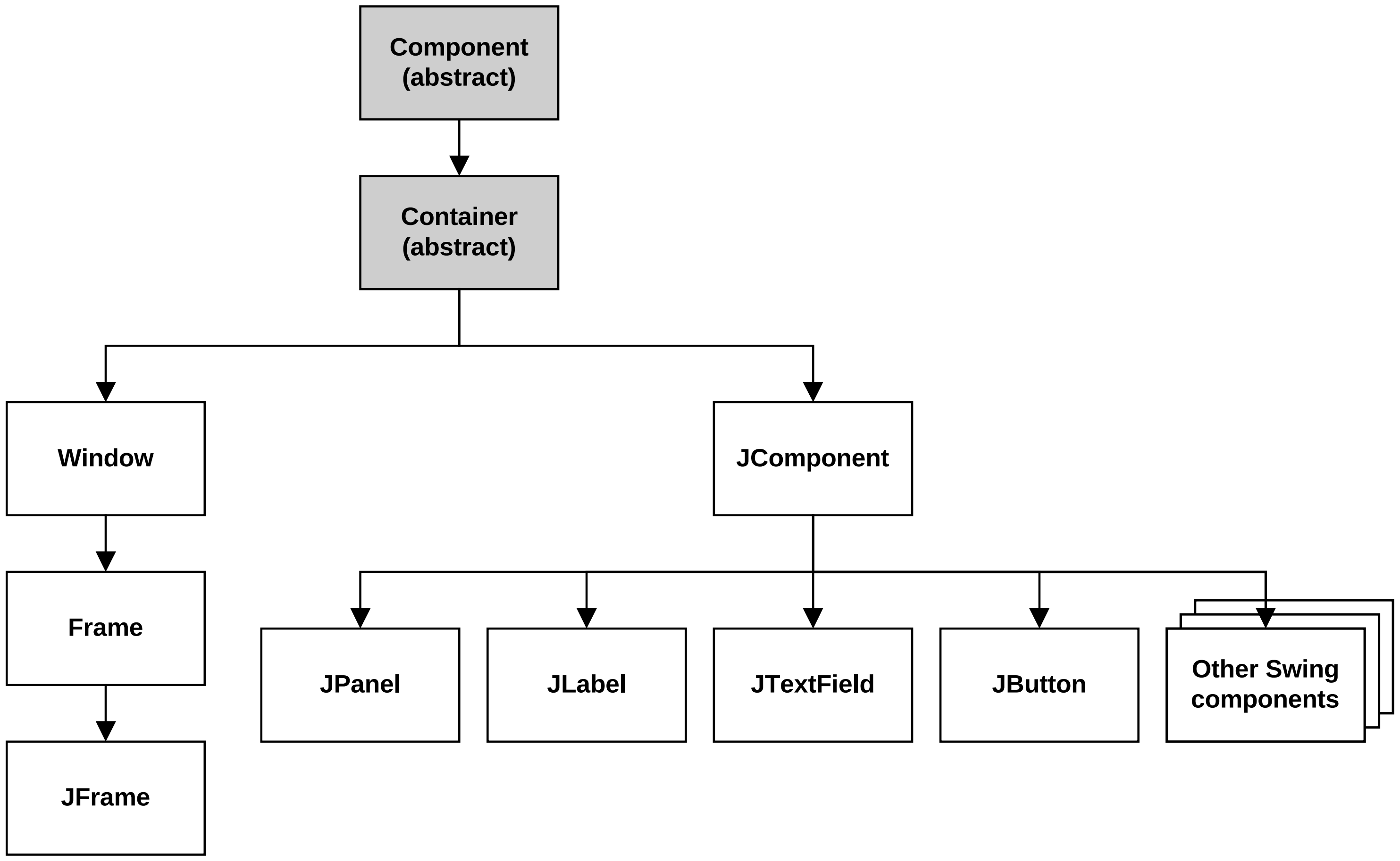
GUI terms

- The window that contains the GUI is called a *frame*.
- The frame contains a *panel* that contains the controls that are displayed by the application.
- The panel in the Future Value Calculator interface contains ten controls: four *labels*, four *text fields*, and two *buttons*.
- Some text fields are not editable. They're used to display output, not to accept user input.

AWT and Swing components

- The *Abstract Window Toolkit* (*AWT*) is an older technology for creating GUIs that look and act a little different on different platforms.
- *Swing* is a newer technology that creates GUIs that are consistent from platform to platform.
- The AWT classes are stored in the java.awt package, while the Swing classes are stored in the javax.swing package. All Swing classes begin with the letter J.

The Component hierarchy



A summary of the classes in the Component hierarchy

Class	Description
Component	An abstract base class that defines any object that can be displayed.
Container	An abstract class that defines any component that can contain other components.
Window	The AWT class that defines a window without a title bar or border.
Frame	The AWT class that defines a window with a title bar and border.
JFrame	The Swing class that defines a window with a title bar and border.
JComponent	A base class for Swing components such as JPanel, JButton, JLabel, and JTextField.

A summary of the classes in the Component hierarchy (continued)

Class	Description
JPanel	The Swing class that defines a panel, which is used to hold other components.
JLabel	The Swing class that defines a label.
JTextField	The Swing class that defines a text field.
JButton	The Swing class that defines a button.

Set methods of the Component class

Method	Description
`setSize(``intWidth, intHeight`**`)`**	Resizes this component using two int values.
`setLocation(``intX, intY`**`)`**	Moves this component to the x and y coordinates specified by two int values.
`setBounds(``intX, intY, intWidth, intHeight`**`)`**	Moves and resizes this component.

Notes

- When you set the location and size of a component, the unit of measurement is *pixels*, which is the number of dots that your monitor uses to display a screen.
- The preferred way to set the location of a component is to use a layout manager.

Set methods of the Component class (continued)

Method	Description
setEnabled(boolean**)**	If the boolean value is true, the component is enabled. If false, the component is disabled, so it doesn't respond to user input or generate events.
setVisible(boolean**)**	Shows this component if the boolean value is true. Otherwise, hides it.
setFocusable(boolean**)**	Determines whether or not this component can receive the focus.
setName(String**)**	Sets the name of this component to the specified string.

Get methods of the Component class

Method	Description
`getHeight()`	Returns the height of this component as an int.
`getWidth()`	Returns the width of this component as an int.
`getX()`	Returns the x coordinate of this component as an int.
`getY()`	Returns the y coordinate of this component as an int.
`getName()`	Returns the name of this component as a String.

Other methods of the Component class

Method	Description
`isEnabled()`	Returns true if the component is enabled.
`isVisible()`	Returns true if the component is visible.
`requestFocusInWindow()`	Moves the focus to the component.

Common methods of the Frame class

Method	Description
`setTitle(String)`	Sets the title to the specified string.
`setResizable(boolean)`	If the boolean value is true, the user can resize the frame.

A class that defines a frame

```
class FutureValueFrame extends JFrame
{
    public FutureValueFrame()
    {
        setTitle("Future Value Calculator");
        setBounds(267, 200, 267, 200);
        setResizable(false);
    }
}
```

A class that displays the frame

```
import javax.swing.*;

public class FutureValueApp
{
    public static void main(String[] args)
    {
        JFrame frame = new FutureValueFrame();
        frame.setVisible(true);
    }
}
```

The setDefaultCloseOperation method of the JFrame class

Method	Description
setDefaultCloseOperation(action**)**	Sets the default close action for the frame.

Constants to set the default close operation

Constant	Description
`JFrame.EXIT_ON_CLOSE`	Exits the application when the user closes the window.
`WindowConstants.DO_NOTHING_ON_CLOSE`	Provides no default action, so the program must explicitly handle the closing event.
`WindowConstants.HIDE_ON_CLOSE`	Hides the frame when the user closes the window. This is the default action.
`WindowConstants.DISPOSE_ON_CLOSE`	Hides and disposes of the frame when the user closes the window.

A class that defines a closeable frame

```
class FutureValueFrame extends JFrame
{
    public FutureValueFrame()
    {
        setTitle("Future Value Calculator");
        setBounds(267, 200, 267, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Two methods of the Toolkit class

Method	Description
`getDefaultToolkit()`	A static method that returns the Toolkit object for the current system.
`getScreenSize()`	Returns the screen resolution as a Dimension object.

Two fields of the Dimension class

Field	Description
`height`	Stores the height of this Dimension object as an int.
`width`	Stores the width of this Dimension object as an int.

How to center a frame using the Toolkit class

- The number of pixels per screen varies depending on the resolution setting of the user’s monitor.
- To determine the number of pixels for the current screen, you can use a Toolkit object, or *toolkit*, to return a Dimension object that contains the number of pixels for the current screen.
- The Toolkit and Dimension classes are in the java.awt package.

A method that centers a frame on the screen

```
private void centerWindow(Window w)
{
    Toolkit tk = Toolkit.getDefaultToolkit();
    Dimension d = tk.getScreenSize();
    setLocation((d.width-w.getWidth())/2,
        (d.height-w.getHeight())/2);
}
```

The constructor for a class that defines a centered frame

```
FutureValueFrame()
{
    setTitle("Future Value Calculator");
    setSize(267, 200);
    centerWindow(this);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

How to add a panel to a frame

- A JFrame object contains several *panes*.
- To add components to a frame, you add them to the *content pane* of the frame.
- A panel is a component that’s used as a container for other components.
- The normal way to build a Swing user interface is to create a panel, add components such as labels, text boxes, and buttons to the panel, then add the panel to the content pane.

Method needed to add components to the content pane with Java 5

Class	Method	Description
JFrame	`add(Component)`	Adds a component to the frame’s content pane.

A JFrame constructor that adds a panel to the content pane with Java 5

```
class FutureValueFrame()
{
    setTitle("Future Value Calculator");
    setSize(267, 200);
    centerWindow(this);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel panel = new JPanel();
    this.add(panel);
}
```

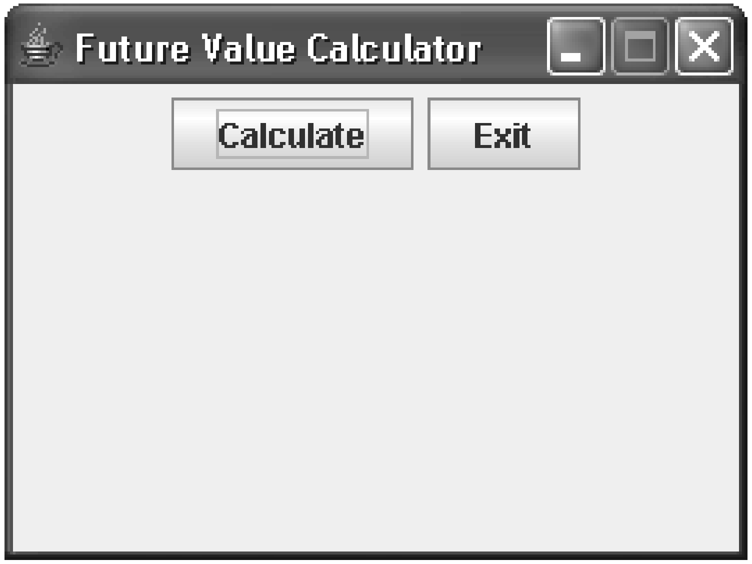
Methods needed to add components to the content pane prior to Java 5

Class	Method	Description
JFrame	**`getContentPane()`**	Returns a Container object that represents the content pane.
Container	**`add(`**`Component`**`)`**	Adds a component (such as a JPanel) to this Container.

Code for adding a panel to the content pane prior to Java 5

```
Container contentPane = this.getContentPane();
contentPane.add(panel);
```

A frame with two buttons



Common constructors of the JButton class

Constructor	Description
`JButton()`	Creates a button with no text.
`JButton(String)`	Creates a button with the text specified by the string.

Common methods of the JButton class

Method	Description
`setText(String)`	Sets the text of the button to the specified string.
`getText()`	Returns a String object for the text of this button.

A JPanel class with two buttons

```
class FuturePanel extends JPanel
{
    private JButton calculateButton;
    private JButton exitButton;

    public FutureValuePanel()
    {
        calculateButton = new JButton("Calculate");
        this.add(calculateButton);
        exitButton = new JButton("Exit");
        this.add(exitButton);
    }
}
```

A frame constructor that adds the panel to the frame

```
class FutureValueFrame()
{
    setTitle("Future Value Calculator");
    setSize(267, 200);
    centerWindow(this);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel panel = new FutureValuePanel();
    this.add(panel);
}
```

How to handle an action event

1. Specify that the class that contains the button implements the ActionListener interface:

```
class FutureValuePanel extends JPanel
                       implements ActionListener
```

2. Add an ActionListener object to the button by calling the addActionListener method:

```
exitButton.addActionListener(this);
```

3. Implement the ActionListener interface by coding the actionPerformed method:

```
public void actionPerformed(ActionEvent e)
{
   Object source = e.getSource();
   if (source == exitButton)
      System.exit(0);
}
```

A panel class that handles two action events

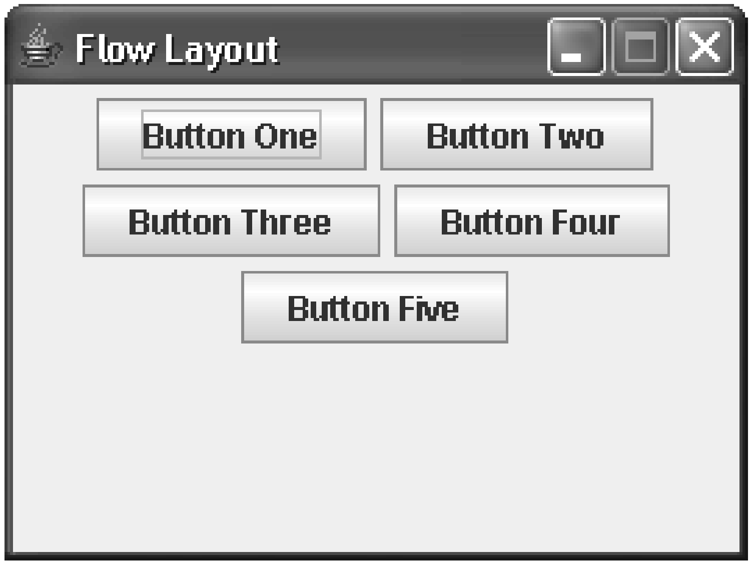
```
class FuturePanel extends JPanel
```

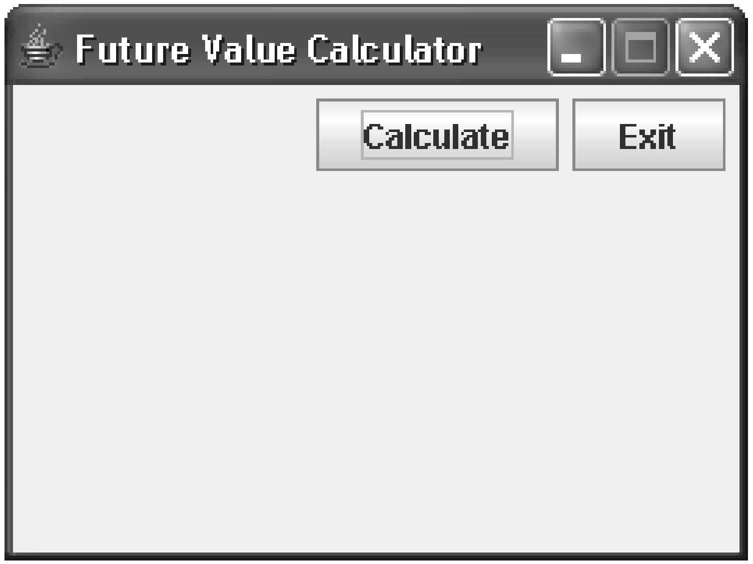
```
class FutureValuePanel extends JPanel
                       implements ActionListener
{
    private JButton calculateButton;
    private JButton exitButton;

    public FutureValuePanel()
    {
        calculateButton = new JButton("Calculate");
        calculateButton.addActionListener(this);
                                  // add an action listener
        this.add(calculateButton);

        exitButton = new JButton("Exit");
        exitButton.addActionListener(this);
                                  // add an action listener
        this.add(exitButton);
    }
```

A panel class that handles two action events (continued)

```
    public void actionPerformed(ActionEvent e)
    {
        Object source = e.getSource();
        if (source == exitButton)
            System.exit(0);
        else if (source == calculateButton)
            calculateButton.setText("Clicked!");
    }
}
```

Two panels that use the Flow layout manager





The setLayout method of the Container class

Method	Description
`setLayout(``LayoutManager`**`)`**	Sets the layout manager for this container.

Two constructors of the FlowLayout class

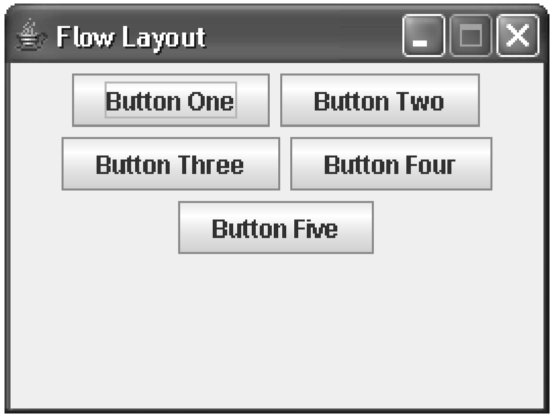
Constructor	Description
`FlowLayout()`	Creates a Flow layout with centered alignment.
`FlowLayout(``alignmentField`**`)`**	Creates a Flow layout with the specified alignment.

Alignment fields of the FlowLayout class

```
CENTER     LEFT     RIGHT
```

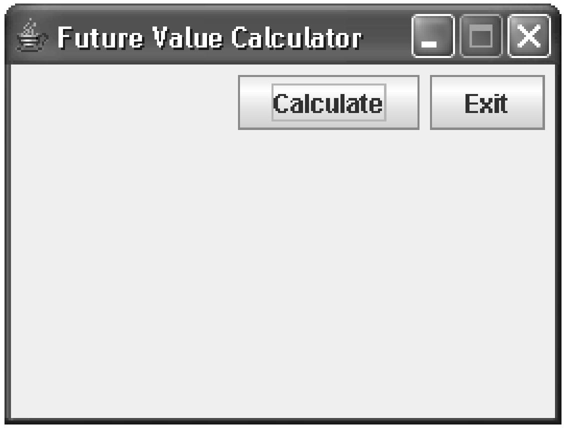
Code that creates the centered button panel

```
class ButtonPanel extends JPanel
{
    public ButtonPanel()
    {
        this.setLayout(
            new FlowLayout(FlowLayout.CENTER));
        this.add(new JButton("Button One"));
        this.add(new JButton("Button Two"));
        this.add(new JButton("Button Three"));
        this.add(new JButton("Button Four"));
        this.add(new JButton("Button Five"));
    }
}
```

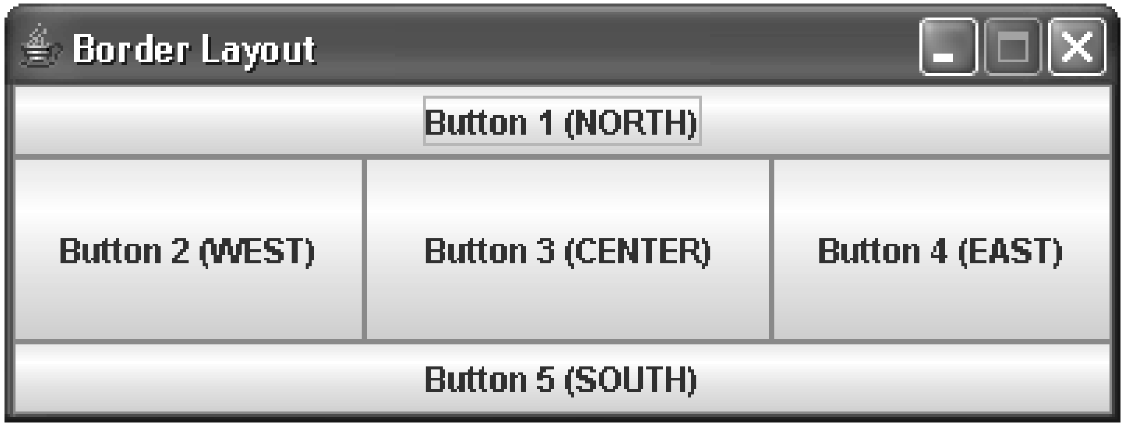


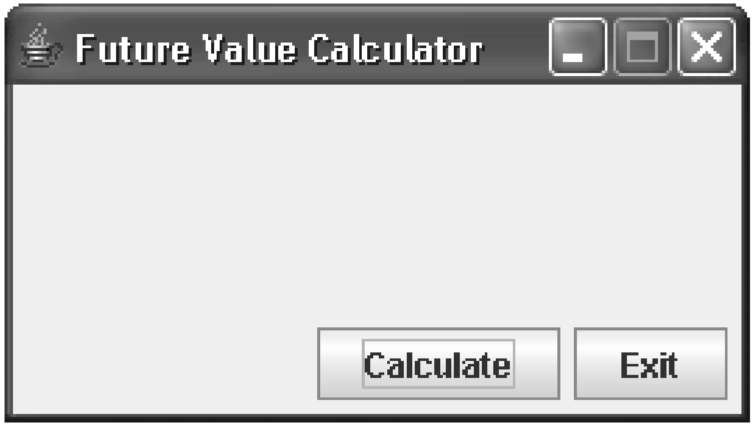
Code that creates the right-aligned button panel

```
class FutureValuePanel extends JPanel
{
    private JButton calculateButton, exitButton;
    public FutureValuePanel()
    {
        this.setLayout(new FlowLayout(FlowLayout.RIGHT));
        calculateButton = new JButton("Calculate");
        this.add(calculateButton);
        exitButton = new JButton("Exit");
        this.add(exitButton);
    }
}
```



Two frames with panels that use the Border layout manager





Common constructor and method of the BorderLayout class

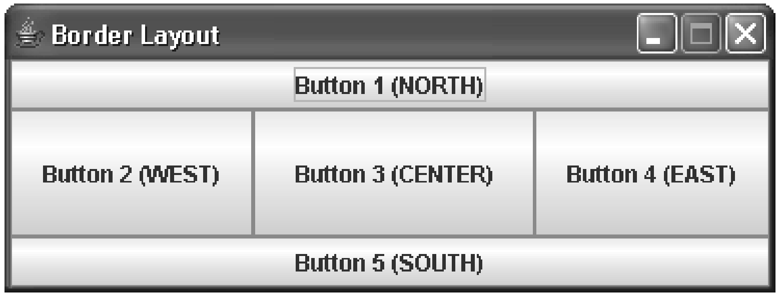
Constructor	Description
`BorderLayout()`	Creates a Border layout manager.
Method	**Description**
`add(``Component, regionField`**`)`**	Adds the component to the specified panel region.

Region fields of the BorderLayout class

`NORTH     WEST     CENTER     EAST     SOUTH`

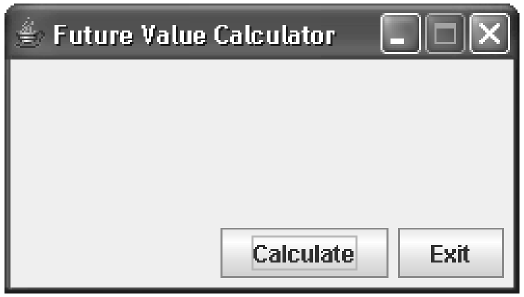
Code that creates the Border Layout frame

```
class BorderLayoutPanel extends JPanel
{
    public BorderLayoutPanel()
    {
        JButton button1 = new JButton("Button 1 (NORTH)");
        JButton button2 = new JButton("Button 2 (WEST)");
        JButton button3 = new JButton("Button 3 (CENTER)");
        JButton button4 = new JButton("Button 4 (EAST)");
        JButton button5 = new JButton("Button 5 (SOUTH)");

        this.setLayout(new BorderLayout());
        this.add(button1, BorderLayout.NORTH);
        this.add(button2, BorderLayout.WEST);
        this.add(button3, BorderLayout.CENTER);
        this.add(button4, BorderLayout.EAST);
        this.add(button5, BorderLayout.SOUTH);
    }
}
```

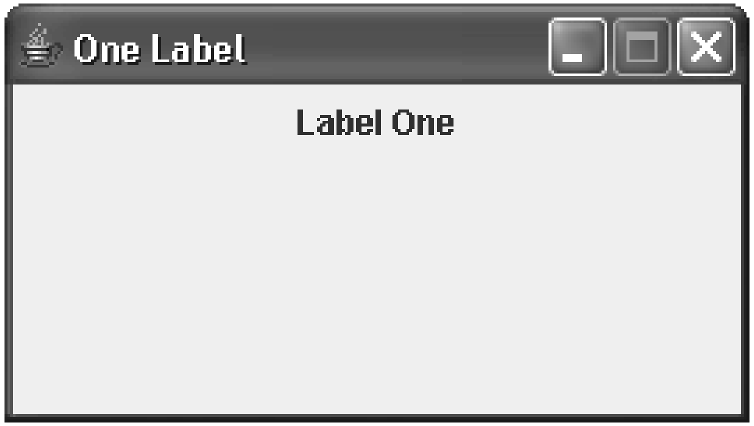


Code that creates the Future Value Calculator frame

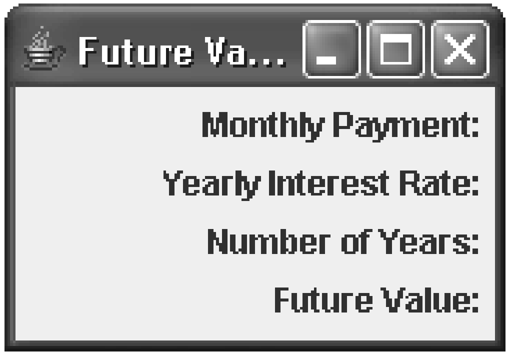
```
class BorderLayoutPanel extends JPanel
{
    public BorderLayoutPanel()
    {
        this.setLayout(new BorderLayout());
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(
            new FlowLayout(FlowLayout.RIGHT));
        JButton calculateButton = new JButton("Calculate");
        JButton exitButton = new JButton("Exit");
        buttonPanel.add(calculateButton);
        buttonPanel.add(exitButton);
        this.add(buttonPanel, BorderLayout.SOUTH);
    }
}
```



Two panels that display labels



Label One



Monthly Payment:
Yearly Interest Rate:
Number of Years:
Future Value:

Common constructors and methods of the JLabel class

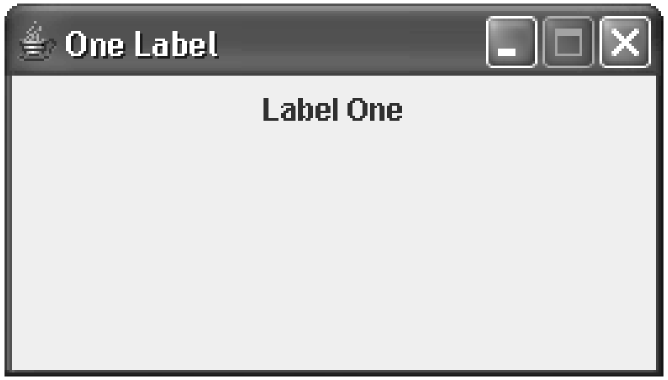
Constructor	Description
`JLabel()`	Creates a blank label.
`JLabel(String)`	Creates a label with the text specified by the string.
Method	**Description**
`getText()`	Returns the text in this text field as a String.
`setText(String)`	Sets the text in this field to the specified string.

How to work with labels

- The JLabel class defines a *label* component that can be used to display text on the panel.
- If you need to refer to a label in code, you can (1) assign it to a variable and then add that variable to the panel or (2) create the label and add it to the panel in a single statement.

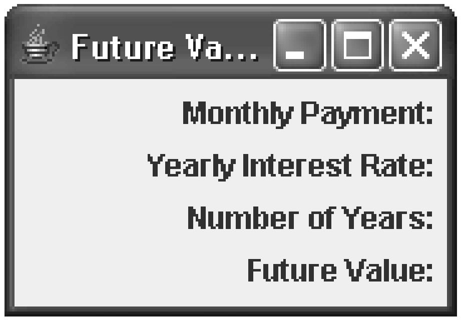
A class that creates the Label One panel

```
class LabelPanel extends JPanel
{
    private JLabel labelOne;

    public LabelPanel()
    {
        labelOne = new JLabel("Label One");
        this.add(labelOne);
    }
}
```

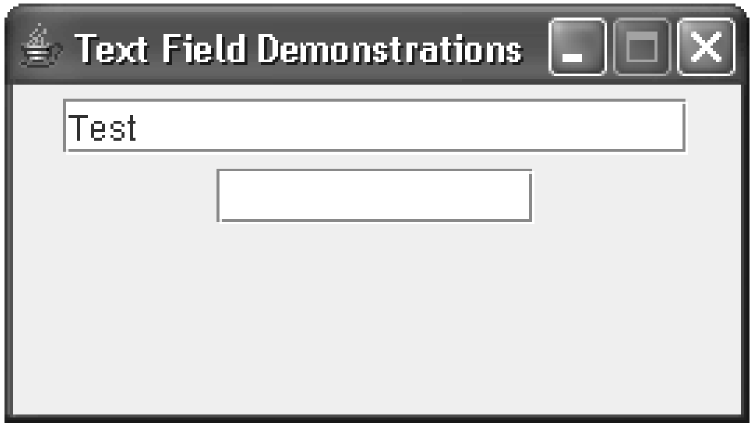


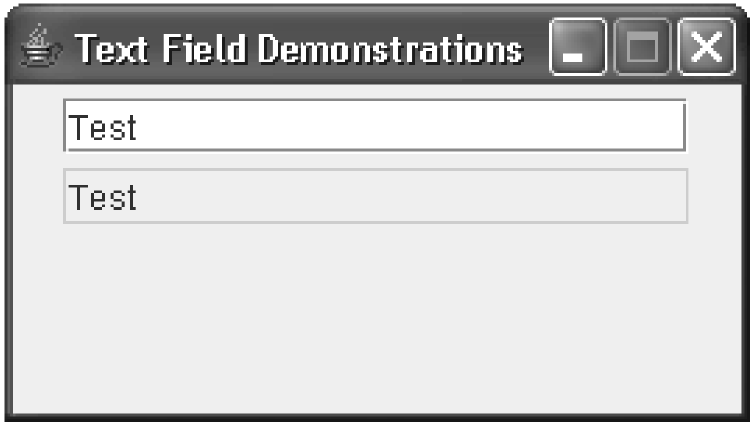
A class that creates the Future Value panel

```
class FutureValuePanel extends JPanel
{
    public FutureValuePanel()
    {
        this.setLayout(new FlowLayout(FlowLayout.RIGHT));
        this.add(new JLabel("Monthly Payment:"));
        this.add(new JLabel("Yearly Interest Rate:"));
        this.add(new JLabel("Number of Years:"));
        this.add(new JLabel("Future Value:"));
    }
}
```



Two versions of a panel that displays two text fields





Common constructors of the JTextField class

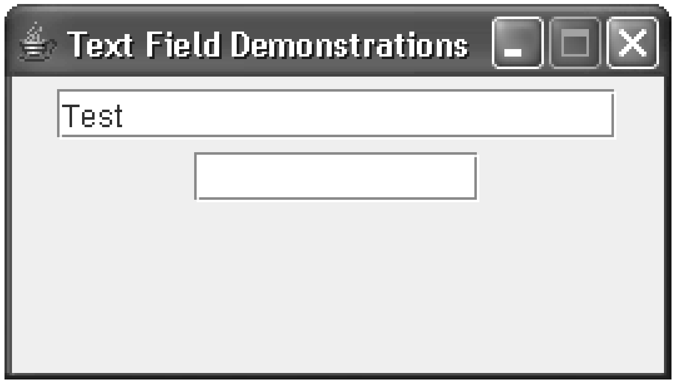
Constructor	Description
`JTextField(``intColumns`**`)`**	Creates a text field with the specified number of columns.
`JTextField(``String, intColumns)`	Creates a text field that starts with the text specified by the string and contains the specified number of columns.

Common methods of the JTextField class

Method	Description
`getText()`	Returns the text in this text field as a String object.
`setText(String)`	Sets the text in this field to the specified string.
`setColumns(intSize)`	Sets the number of columns to the specified value.
`setEditable(boolean)`	Determines whether or not the field can be edited.
`setFocusable(boolean)`	Determines whether or not the field can receive the focus.

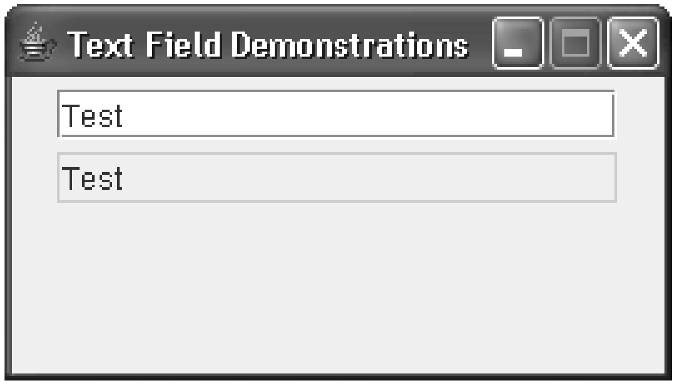
A class that creates the panel with two text fields

```
class TextFieldPanel extends JPanel
{
    private JTextField textFieldOne, textFieldTwo;

    public TextFieldPanel()
    {
        textFieldOne = new JTextField("Test", 20);
        this.add(textFieldOne);
        textFieldTwo = new JTextField(10);
        this.add(textFieldTwo);
    }
}
```

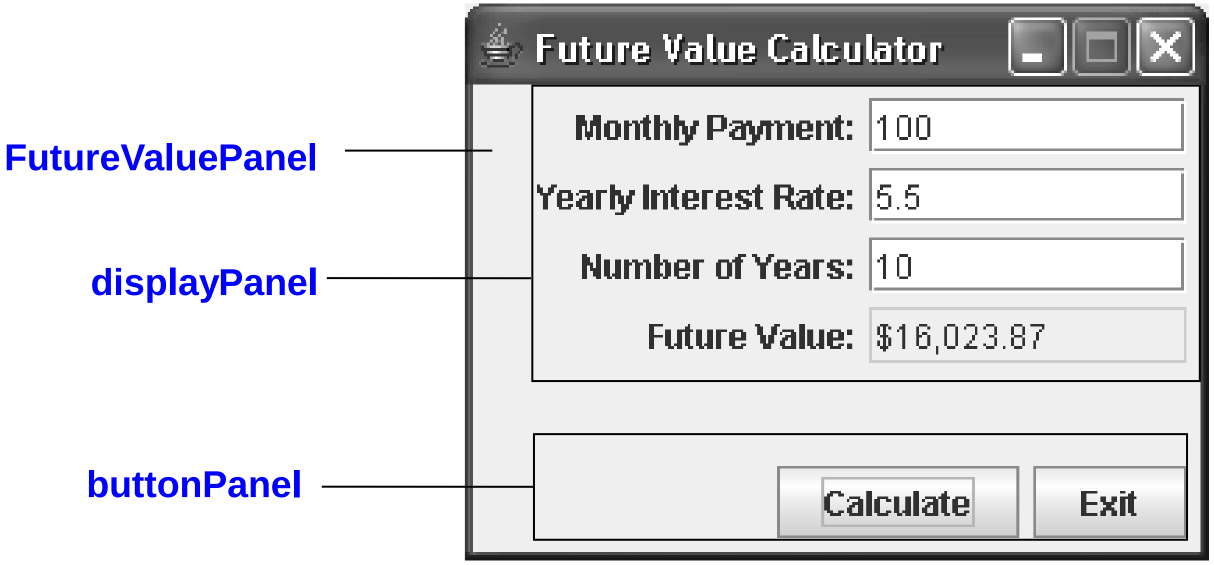


Code that modifies the second text field

```
public void modifyFields()
{
    String data = textFieldOne.getText();
    textFieldTwo.setText(data);
    textFieldTwo.setColumns(20);
    textFieldTwo.setEditable(false);
}
```



The panels of the user interface for the Future Value Calculator application



Monthly Payment: 100
Yearly Interest Rate: 5.5
Number of Years: 10
Future Value: $16,023.87

The Future Value Calculator application

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.text.*;

public class FutureValueApp
{
    public static void main(String[] args)
    {
        JFrame frame = new FutureValueFrame();
        frame.setVisible(true);
    }
}
```

The Future Value Calculator application (cont.)

```
class FutureValueFrame extends JFrame
{
    public FutureValueFrame()
    {
        setTitle("Future Value Calculator");
        setSize(267, 200);
        centerWindow(this);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panel = new FutureValuePanel();
        this.add(panel);
    }

    private void centerWindow(Window w)
    {
        Toolkit tk = Toolkit.getDefaultToolkit();
        Dimension d = tk.getScreenSize();
        setLocation((d.width-w.getWidth())/2,
            (d.height-w.getHeight())/2);
    }
}
```

The Future Value Calculator application (cont.)

```
class FuturevaluePanel extends JPanel
                       implements ActionListener
{
    private JTextField  paymentTextField,
                        rateTextField,
                        yearsTextField,
                        futureValueTextField;
    private JLabel      paymentLabel,
                        rateLabel,
                        yearsLabel,
                        futureValueLabel;
    private JButton     calculateButton,
                        exitButton;

    public FutureValuePanel()
    {
        // display panel
        JPanel displayPanel = new JPanel();
        displayPanel.setLayout(
            new FlowLayout(FlowLayout.RIGHT));
```

The Future Value Calculator application (cont.)

```
// payment label
paymentLabel = new JLabel("Monthly Payment:");
displayPanel.add(paymentLabel);

// payment text field
paymentTextField = new JTextField(10);
displayPanel.add(paymentTextField);

// rate label
rateLabel = new JLabel("Yearly Interest Rate:");
displayPanel.add(rateLabel);

// rate text field
rateTextField = new JTextField(10);
displayPanel.add(rateTextField);

// years label
yearsLabel = new JLabel("Number of Years:");
displayPanel.add(yearsLabel);
```

The Future Value Calculator application (cont.)

```
// years text field
yearsTextField = new JTextField(10);
displayPanel.add(yearsTextField);

// future value label
futureValueLabel = new JLabel("Future Value:");
displayPanel.add(futureValueLabel);

// future value text field
futureValueTextField = new JTextField(10);
futureValueTextField.setEditable(false);
futureValueTextField.setFocusable(false);
displayPanel.add(futureValueTextField);

// button panel
JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(
    new FlowLayout(FlowLayout.RIGHT));
```

The Future Value Calculator application (cont.)

```
        // calculate button
        calculateButton = new JButton("Calculate");
        calculateButton.addActionListener(this);
        buttonPanel.add(calculateButton);

        // exit button
        exitButton = new JButton("Exit");
        exitButton.addActionListener(this);
        buttonPanel.add(exitButton);

        // add panels to main panel
        this.setLayout(new BorderLayout());
        this.add(displayPanel, BorderLayout.CENTER);
        this.add(buttonPanel, BorderLayout.SOUTH);
    }
```

The Future Value Calculator application (cont.)

```
    public void actionPerformed(ActionEvent e)
    {   Object source = e.getSource();
        if (source == exitButton)
            System.exit(0);
        else if (source == calculateButton)
        {   double payment = Double.parseDouble(
                paymentTextField.getText());
            double rate = Double.parseDouble(
                rateTextField.getText());
            int years = Integer.parseInt(
                yearsTextField.getText());
            double futureValue =
                FinancialCalculations.calculateFutureValue(
                payment, rate, years);
            NumberFormat currency =
                NumberFormat.getCurrencyInstance();
            futureValueTextField.setText(
                currency.format(futureValue));
        }
    }
}
```