

# **7 Abstract Windowing Toolkit and Swing**



# Topics

- Abstract Windowing Toolkit (AWT) vs. Swing
- AWT GUI Components
  - Fundamental Window Classes
  - *Graphics*
  - More AWT Components



# Topics

- Layout Managers
  - The FlowLayout Manager
  - The BorderLayout Manager
  - The GridLayout Manager
  - Panels and Complex Layouts
- Swing GUI Components
  - Setting Up Top-Level Containers
  - A JFrame Example
  - A JOptionPane Example



# Abstract Windowing Toolkit (AWT) vs. Swing

- Similarities:
  - Tools provided by Java for developing interactive GUI applications
  - Provides GUI components that can be used in creating Java applications and applets
- Java Foundation Classes (JFCs)
  - Important part of the Java SDK
  - Collection of APIs that simplifies development Java GUI applications
  - Primarily consists of five APIs
    - AWT
    - Swing
    - Java2D
    - Accessibility
    - Drag and Drop



# Abstract Windowing Toolkit (AWT) vs. Swing

- AWT
  - Some AWT components use native code
  - Platform-dependent
  - Ensure that the look and feel of an application run on different machines be comparable
- Swing
  - Written entirely using the Java programming language
  - Platform-independent
  - Ensures applications deployed across different platforms have the same appearance
  - Built around a number of APIs that implement various parts of the AWT
    - Can be used with AWT



# AWT GUI Components: Fundamental Window Classes

- GUI components such as buttons or text fields are placed in containers.

<b><i>AWT Class</i></b>	<b><i>Description</i></b>
Component	An abstract class for objects that can be displayed on the console and interact with the user. The root of all other AWT classes.
Container	An abstract subclass of the <i>Component</i> class. A component that can contain other components.
Panel	Extends the <i>Container</i> class. A frame or window without the titlebar, the menubar nor the border. Superclass of the <i>Applet</i> class.
Window	Also extends <i>Container</i> class. A top-level window, which means that it cannot be contained in any other object. Has no borders and no menubar.
Frame	Extends the <i>Window</i> class. A window with a title, menubar, border, and resizing corners. Has four constructors, two of which have the following signatures:  <code>Frame()</code>  <code>Frame(String title)</code>



# AWT GUI Components: Window Class Methods

- Setting the size of the window:

```
void setSize(int width, int height)
```

```
void setSize(Dimension d)
```

- *Dimension d* has *width* and *height* as fields

- A window by default is not visible. Setting its visibility:

```
void setVisible(boolean b)
```

- If *b* is *true*, window is set to be visible



# AWT GUI Components: Fundamental Window Classes

- *Frame* objects are usually used in designing GUI applications

```
1 import java.awt.*;
2 /* Try the available buttons in the frame */
3 public class SampleFrame extends Frame {
4     public static void main(String args[]) {
5         SampleFrame sf = new SampleFrame();
6         sf.setSize(100, 100); //Try removing this
7         sf.setVisible(true);  //Try removing this
8     }
9 }
```





# AWT GUI Components: *Graphics*

- *Graphics* class (*abstract*) methods:

drawLine()	drawPolyline()	setColor()
fillRect()	drawPolygon()	getFont()
drawRect()	fillPolygon()	setFont()
clearRect()	getColor()	drawString()

- *Color* class constructors

<b>Constructor Format</b>	<b>Description</b>
Color(int r, int g, int b)	Integer value is from 0 to 255.
Color(float r, float g, float b)	Float value is from 0.0 to 1.0.
Color(int rgbValue)	Value range from 0 to $2^{24}-1$ (black to white). Red: bits 16-23 Green: bits 8-15 Blue: bits 0-7



# AWT GUI Components:

## *Graphics Example*

```
1 import java.awt.*;
2 public class GraphicPanel extends Panel {
3     GraphicPanel() {
4         setBackground(Color.black);
5     }
6     public void paint(Graphics g) {
7         g.setColor(new Color(0,255,0)); //green
8         g.setFont(new Font("Helvetica",Font.PLAIN,16));
9         g.drawString("Hello GUI World!", 30, 100);
10        g.setColor(new Color(1.0f,0,0)); //red
11        g.fillRect(30, 100, 150, 10);
12    }
13 //continued...
```



# AWT GUI Components: *Graphics Example*

```
14  /* need to place Panel in Frame or other Window */
15  public static void main(String args[]) {
16      Frame f = new Frame("Testing Graphics Panel");
17      GraphicPanel gp = new GraphicPanel();
18      f.add(gp);
19      f.setSize(600, 300);
20      f.setVisible(true);
21  }
22 }
```



# More AWT Components

- AWT controls
  - Components that allow the user to interact with a GUI application
  - Subclasses of the component class

Label	Button	Choice
TextField	Checkbox	List
TextArea	CheckboxGroup	Scrollbar



# More AWT Components: Example

```
1 import java.awt.*;
2 class FrameWControls extends Frame {
3     public static void main(String args[]) {
4         FrameWControls fwc = new FrameWControls();
5         fwc.setLayout(new FlowLayout());
6         fwc.setSize(600, 600);
7         fwc.add(new Button("Test Me!"));
8         fwc.add(new Label("Labe"));
9         fwc.add(new TextField());
10        CheckboxGroup cbg = new CheckboxGroup();
11        fwc.add(new Checkbox("chk1", cbg, true));
12        fwc.add(new Checkbox("chk2", cbg, false));
13        fwc.add(new Checkbox("chk3", cbg, false));
```

```
14 //continued...
```

Introduction to Programming 2



# More AWT Components: Example

```
15      List list = new List(3, false);
16      list.add("MTV");
17      list.add("V");
18      fwc.add(list);
19      Choice chooser = new Choice();
20      chooser.add("Avril");
21      chooser.add("Monica");
22      chooser.add("Britney");
23      fwc.add(chooser);
24      fwc.add(new Scrollbar());
25      fwc.setVisible(true);
26  }
27 }
```



# Layout Managers

- Definition:
  - Determines the position and size of the components within a container
  - Governs the layout of the components in the container
- Some of the layout managers in Java
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GridBagLayout
  - CardLayout



# Layout Managers: Methods

- Setting the layout manager:

```
void setLayout (LayoutManager mgr)
```

- Can pass *null*, no layout manager in use

- If no layout manager is used, need to position the elements manually

```
public void setBounds (int x, int y, int width, int height)
```

- Method of the *Component* class
- Quite difficult and tedious if you have several *Component* objects
  - Need to call this method for each object





# Layout Managers: The *FlowLayout* Manager

- Default manager for the *Panel* class and its subclasses
  - The *Applet* class is a subclass of *Panel*
- Positions the components in a left to right and top to bottom manner, starting at the upper-lefthand corner
  - Typing using a word editor



# Layout Managers: The *FlowLayout* Manager

- Has three constructors:

## ***FlowLayout Constructors***

`FlowLayout()`

Creates a new `FlowLayout` object with the center alignment and 5-unit horizontal and vertical gap applied to the components by default.

`FlowLayout(int align)`

Creates a new `FlowLayout` object with the specified alignment and the default 5-unit horizontal and vertical gap applied to the components.

`FlowLayout(int align, int hgap, int vgap)`

Creates a new `FlowLayout` object with the first argument as the alignment applied and the *hgap*-unit horizontal and *vgap*-unit vertical gap applied to the components.



# Layout Managers: The *FlowLayout* Manager

- Gap
  - Spacing between the components
  - Measured in pixels

- Possible alignment values:

`FlowLayout.LEFT`

`FlowLayout.CENTER`

`FlowLayout.RIGHT`



# Layout Managers: The *FlowLayout* Manager

```
1 import java.awt.*;
2 class FlowLayoutDemo extends Frame {
3     public static void main(String args[]) {
4         FlowLayoutDemo fld = new FlowLayoutDemo();
5         fld.setLayout(new FlowLayout(FlowLayout.RIGHT,
6                                     10, 10));
7         fld.add(new Button("ONE"));
8         fld.add(new Button("TWO"));
9         fld.add(new Button("THREE"));
10        fld.setSize(100, 100);
11        fld.setVisible(true);
12    }
13 }
```



# Layout Managers: The *FlowLayout* Manager

- Sample output:



# Layout Managers:

## The *BorderLayout* Manager

- Default layout for *Window* objects and its subclasses
  - Includes those of *Frame* and *Dialog* type
- Divides *Container* object into five parts where *Component* objects are added
  - North - stretch horizontally
  - South - stretch horizontally
  - East - adjust vertically
  - West - adjust vertically
  - Center - adjusts in both directions



# Layout Managers:

## The *BorderLayout* Manager

- Has two constructors

<b><i>BorderLayout</i> Constructors</b>
<code>BorderLayout()</code>
Creates a new <code>BorderLayout</code> object with no spacing applied among the different components.
<code>BorderLayout(int hgap, int vgap)</code>
Creates a new <code>BorderLayout</code> object with <i>hgap</i> -unit horizontal and <i>vgap</i> -unit spacing applied among the different components.

- Parameters *hgap* and *vgap* refers to the spacing between the components within the container



# Layout Managers:

## The *BorderLayout* Manager

- Adding a component to a specified region:
  - Use the *add* method and pass two arguments:
    - Component to add
    - Region where the component is to be positioned
  - Only one component can be placed in one region
- Valid regions:
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.CENTER





# Layout Managers:

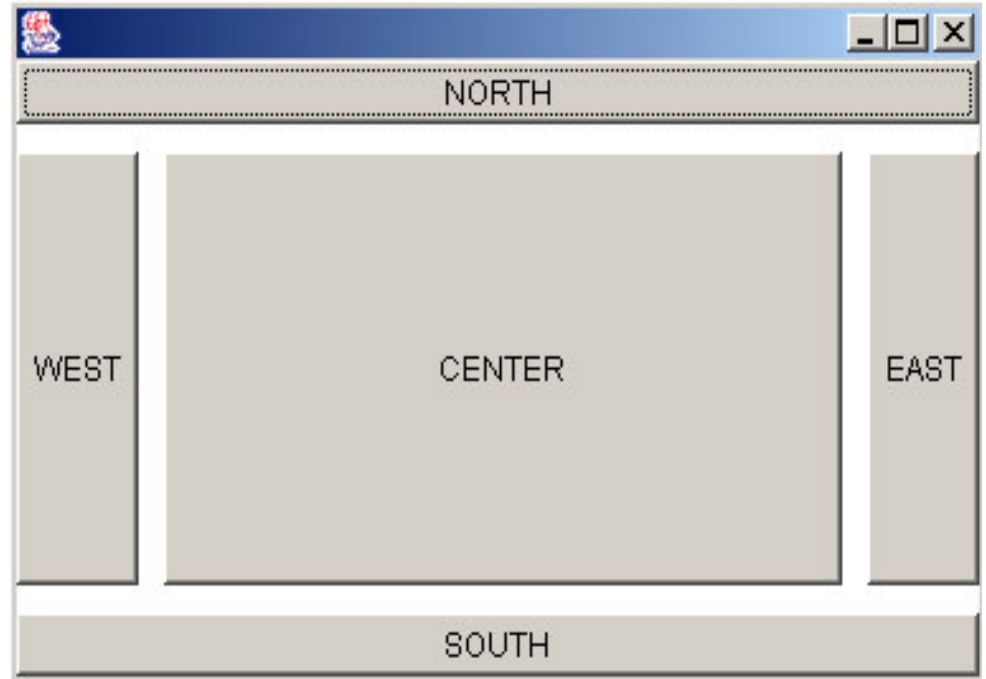
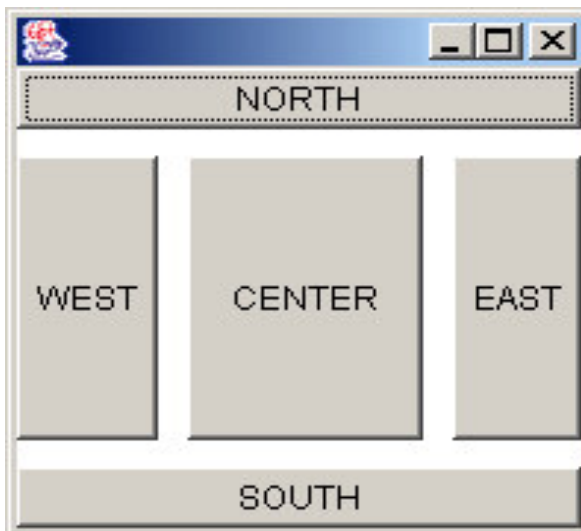
## The *BorderLayout* Manager

```
1 import java.awt.*;
2 class BorderLayoutDemo extends Frame {
3     public static void main(String args[]) {
4         BorderLayoutDemo bld = new BorderLayoutDemo();
5         bld.setLayout(new BorderLayout(10, 10));
6         bld.add(new Button("NORTH"), BorderLayout.NORTH);
7         bld.add(new Button("SOUTH"), BorderLayout.SOUTH);
8         bld.add(new Button("EAST"), BorderLayout.EAST);
9         bld.add(new Button("WEST"), BorderLayout.WEST);
10        bld.add(new Button("CENTER"), BorderLayout.CENTER);
11        bld.setSize(200, 200);
12        bld.setVisible(true);
13    }
14 }
```



# Layout Managers: The *BorderLayout* Manager

- Sample output:
- After resizing:



# Layout Managers: The *GridLayout* Manager

- Like FlowLayout
  - Positions components from left to right and top to bottom
  - Starts adding components at the upper-lefthand corner
- Divides the container into a number of rows and columns
  - Regions are equally sized
  - Ignores the component's preferred size



# Layout Managers:

## The *GridLayout* Manager

- Has the following constructors:

<b><i>GridLayout Constructors</i></b>
<code>GridLayout()</code>
Creates a new <code>GridLayout</code> object with a single row and a single column by default.
<code>GridLayout(int rows, int cols)</code>
Creates a new <code>GridLayout</code> object with the specified number of rows and columns.
<code>GridLayout(int rows, int cols, int hgap, int vgap)</code>
Creates a new <code>GridLayout</code> object with the specified number of rows and columns. <i>hgap</i> -unit horizontal and <i>vgap</i> -unit vertical spacings are applied to the components.



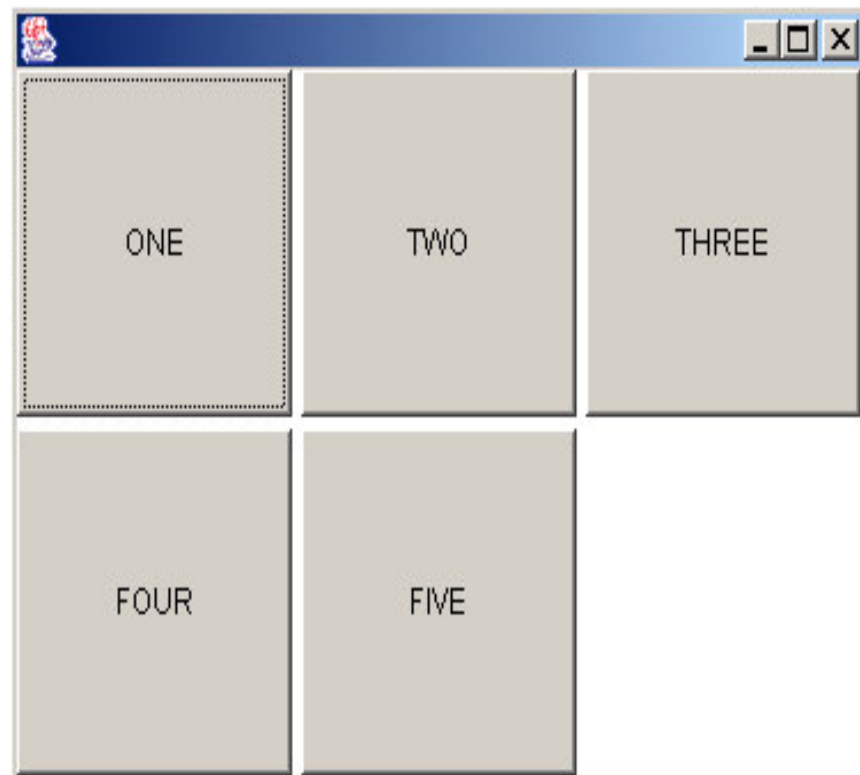
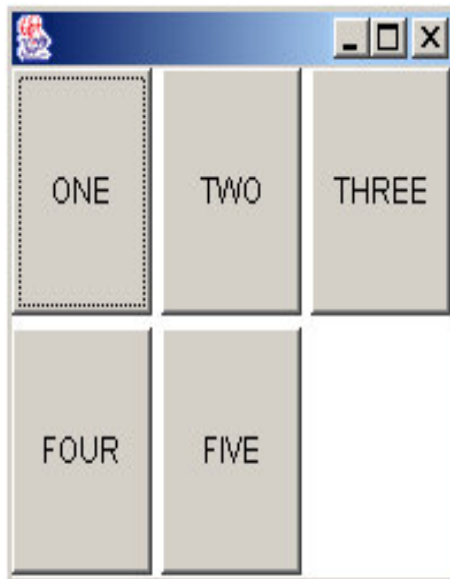
# Layout Managers: The *GridLayout* Manager

```
1 import java.awt.*;
2 class GridLayoutDemo extends Frame {
3     public static void main(String args[]) {
4         GridLayoutDemo gld = new GridLayoutDemo();
5         gld.setLayout(new GridLayout(2, 3, 4, 4));
6         gld.add(new Button("ONE"));
7         gld.add(new Button("TWO"));
8         gld.add(new Button("THREE"));
9         gld.add(new Button("FOUR"));
10        gld.add(new Button("FIVE"));
11        gld.setSize(200, 200);
12        gld.setVisible(true);
13    }
14 }
```



# Layout Managers: The *GridLayout* Manager

- Sample output:
- After resizing:



# Panels and Complex Layouts

- For more complex layouts
  - Can combine the different layout managers
  - Use of panels at the same time
- Recall:
  - A *Panel* is a *Container* and a *Component*
  - Can insert *Components* into the *Panel*
  - Can add *Panel* to a *Container*



# Panels and Complex Layouts

```
1 import java.awt.*;
2 class ComplexLayout extends Frame {
3     public static void main(String args[]) {
4         ComplexLayout cl = new ComplexLayout();
5         Panel panelNorth = new Panel();
6         Panel panelCenter = new Panel();
7         Panel panelSouth = new Panel();
8         /* North Panel */
9         //Panels use FlowLayout by default
10        panelNorth.add(new Button("ONE"));
11        panelNorth.add(new Button("TWO"));
12        panelNorth.add(new Button("THREE"));
13 //continued...
```





# Panels and Complex Layouts

```
14      /* Center Panel */
15      panelCenter.setLayout(new GridLayout(4,4));
16      panelCenter.add(new TextField("1st"));
17      panelCenter.add(new TextField("2nd"));
18      panelCenter.add(new TextField("3rd"));
19      panelCenter.add(new TextField("4th"));
20      /* South Panel */
21      panelSouth.setLayout(new BorderLayout());
22      panelSouth.add(new Checkbox("Choose me!",
                                   BorderLayout.CENTER);
23      panelSouth.add(new Checkbox("I'm here!",
                                   BorderLayout.EAST);
24      panelSouth.add(new Checkbox("Pick me!",
                                   BorderLayout.WEST);
25
26      //continued...
```



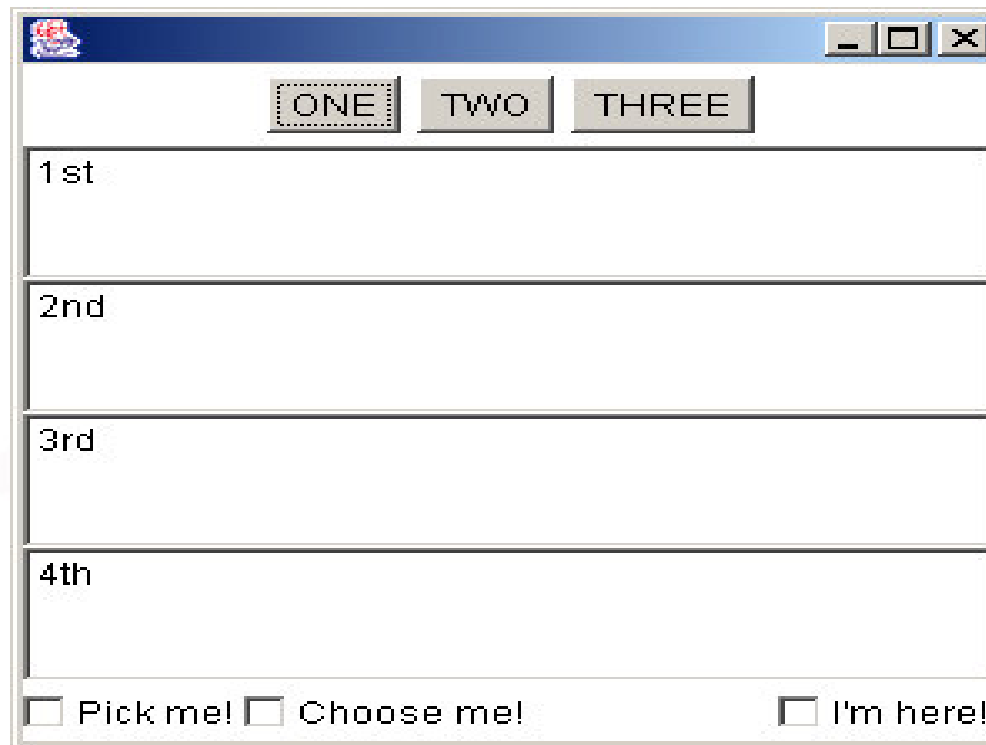
# Panels and Complex Layouts

```
28      /* Adding the Panels to the Frame container */
29      //Frames use BorderLayout by default
30      cl.add(panelNorth, BorderLayout.NORTH);
31      cl.add(panelCenter, BorderLayout.CENTER);
32      cl.add(panelSouth, BorderLayout.SOUTH);
33      cl.setSize(300,300);
34      cl.setVisible(true);
35  }
36 }
```



# Panels and Complex Layouts

- Sample output:



A sample GUI window titled "JEDI" with a blue title bar and standard window controls (minimize, maximize, close). The window contains the following elements:

- Three buttons labeled "ONE", "TWO", and "THREE" arranged horizontally at the top.
- Four text input fields stacked vertically, each preceded by a label: "1st", "2nd", "3rd", and "4th".
- Three checkboxes at the bottom: "Pick me!", "Choose me!", and "I'm here!".

# Swing GUI Components

- Package is found in *javax.swing*
- Written entirely using Java
  - Have the same look and feel even when executed on different platforms
- Provides more interesting components
  - Color chooser
  - Option pane



# Swing GUI Components

- Names of the Swing GUI components are almost similar to that of AWT
  - Name of AWT components but prefixed with J
  - Example:
    - AWT: *Button* class
    - Corresponding Swing component: *JButton* class



# Swing GUI Components

<b><i>Swing Component</i></b>	<b><i>Description</i></b>
JComponent	The root class for all Swing components, excluding top-level containers.
JButton	A "push" button. Corresponds to the <i>Button</i> class in the AWT package.
JCheckBox	An item that can be selected or deselected by the user. Corresponds to the <i>Checkbox</i> class in the AWT package.
JFileChooser	Allows user to select a file. Corresponds to the <i>FileChooser</i> class in the AWT package.
JTextField	Allows for editing of a single-line text. Corresponds to <i>TextField</i> class in the AWT package.
JFrame	Extends and corresponds to the <i>Frame</i> class in the AWT package but the two are slightly incompatible in terms of adding components to this container. Need to get the current content pane before adding a component.
JPanel	Extends <i>JComponent</i> . A simple container class but not top-level. Corresponds to <i>Panel</i> class in the AWT package.



# Swing GUI Components

<b><i>Swing Component</i></b>	<b><i>Description</i></b>
JApplet	Extends and corresponds to the <i>Applet</i> class in the AWT package. Also slightly incompatible with the <i>Applet</i> class in terms of adding components to this container.
JOptionPane	Extends <i>JComponent</i> . Provides an easy way of displaying pop-up dialog box.
JDialog	Extends and corresponds to the <i>Dialog</i> class in the AWT package. Usually used to inform the user of something or prompt the user for an input.
JColorChooser	Extends <i>JComponent</i> . Allow the user to select a color.



# Swing: Setting Up Top-Level Containers

- Top-level containers in Swing are slightly incompatible with those in AWT
  - In terms of adding components to the container
- Adding a component to the container:
  - Get the content pane of the container
    - Use the *getContentPane* method
  - Add components to the content pane
    - Still use the *add* method





# Swing: A *JFrame* Example

```
1 import javax.swing.*;
2 import java.awt.*;
3 class SwingDemo {
4     JFrame frame;
5     JPanel panel;
6     JTextField textField;
7     JButton button;
8     Container contentPane;
9     public static void main(String args[]) {
10         SwingDemo sd = new SwingDemo();
11         sd.launchFrame();
12     }
13 //continued...
```



# Swing: A *JFrame* Example

```
14 void launchFrame() {  
15     /* initialization */  
16     frame = new JFrame("My First Swing Application");  
17     panel = new JPanel();  
18     textField = new JTextField("Default text");  
19     button = new JButton("Click me!");  
20     contentPane = frame.getContentPane();  
21     //add components to panel-FlowLayout by default  
22     panel.add(textField);  
23     panel.add(button);  
24     /* add components to contentPane- BorderLayout */  
25     contentPane.add(panel, BorderLayout.CENTER);  
26     frame.pack(); //Size of frame based on components  
27     frame.setVisible(true);  
28 }  
29 }
```



# Swing: A *JFrame* Example

- The *java.awt* package is still imported
  - The layout managers in use are defined in this package
  - Giving a title to the frame and packing the components within the frame is applicable for AWT frames too
- Coding convention:
  - Declare components as fields
  - A *launchFrame* method is defined:
    - Initialization and addition of components
  - No longer just extend the *Frame* class
  - Advantage: organized and easier to add event handling codes



# Swing: A *JFrame* Example

- Sample output:



# Swing: A *JOptionPane* Example

```
1 import javax.swing.*;
2 class JOptionPaneDemo {
3     JOptionPane optionPane;
4     void launchFrame() {
5         optionPane = new JOptionPane();
6         String name = optionPane.showInputDialog(
7             "Hi, what's your name?");
8         optionPane.showMessageDialog(null,
9             "Nice to meet you, " + name + ".",
10            "Greeting...",optionPane.PLAIN_MESSAGE);
11         System.exit(0);
12     }
13     public static void main(String args[]) {
14         new JOptionPaneDemo().launchFrame();    }    }
```



# Swing: A *JFrame* Example

- Sample output:



# Summary

- Abstract Windowing Toolkit (AWT) vs. Swing
  - Similarities
  - Differences
- AWT GUI Components
  - Fundamental Window Classes
    - *Component, Container, Window, Frame, Panel*
  - *Graphics*
    - Methods and the *Color* class
  - More AWT Components
    - *Label, TextField, TextArea, Button, Checkbox, CheckboxGroup, Choice, List, Scrollbar*



# Summary

- Layout Managers
  - The *FlowLayout* Manager
  - The *BorderLayout* Manager
  - The *GridLayout* Manager
  - Creating Complex Layouts
- Swing GUI Components
  - Setting Up Top-Level Containers
    - Use *getContentPane* method
    - Use *add* method
  - *JFrame*, *JOptionPane*

