GUI basics

Giuseppe Narzisi
V22.0102-003
Spring 2009
Creating GUI Objects

// Create a button with text OK
JButton jbtOK = new JButton("OK");

// Create a label with text "Enter your name: 
JLabel jlblName = new JLabel("Enter your name: ");

// Create a text field with text "Type Name Here"
JTextField jtfName = new JTextField("Type Name Here");

// Create a check box with text bold
JCheckBox jchkBold = new JCheckBox("Bold");

// Create a radio button with text red
JRadioButton jrbRed = new JRadioButton("Red");

// Create a combo box with choices red, green, and blue
JComboBox jcboColor = new JComboBox(new String[]{"Red", "Green", "Blue"});
Frames

- Frame is a window that is not contained inside another window. Frame is the basis to contain other user interface components in Java GUI applications.
- The JFrame class can be used to create windows.
- For Swing GUI programs, use JFrame class to create windows.
import javax.swing.*;
public class MyFrame {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Test Frame");
        frame.setSize(400, 300);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
Adding Components into a Frame

// Add a button into the frame
frame.getContentPane().add(new JButton("OK"));
Content Pane Delegation in JDK 1.5

```java
// Add a button into the frame
frame.getContentPane().add(new JButton("OK"));
```

```java
// Add a button into the frame
frame.add(new JButton("OK"));
```
# JFrame Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFrame()</td>
<td>Creates a default frame with no title.</td>
</tr>
<tr>
<td>JFrame(title: String)</td>
<td>Creates a frame with the specified title.</td>
</tr>
<tr>
<td>getSize(width: int, height: int): void</td>
<td>Specifies the size of the frame.</td>
</tr>
<tr>
<td>setLocation(x: int, y: int): void</td>
<td>Specifies the upper-left corner location of the frame.</td>
</tr>
<tr>
<td>setVisible(visible: boolean): void</td>
<td>Sets true to display the frame.</td>
</tr>
<tr>
<td>setDefaultCloseOperation(mode: int): void</td>
<td>Specifies the operation when the frame is closed.</td>
</tr>
<tr>
<td>setLocationRelativeTo (c: Component): void</td>
<td>Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.</td>
</tr>
</tbody>
</table>
Using Panels as Sub-Containers

• Panels act as sub-containers for grouping user interface components.

• It is recommended that you place the user interface components in panels and place the panels in a frame. You can also place panels in a panel.

• To add a component to JFrame, you actually add it to the content pane of JFrame. To add a component to a panel, you add it directly to the panel using the add method.
Creating a JPanel

You can use `new JPanel()` to create a panel with a default FlowLayout manager or `new JPanel(LayoutManager)` to create a panel with the specified layout manager. Use the `add(Component)` method to add a component to the panel. For example:

```java
JPanel p = new JPanel();
p.add(new JButton("OK"));
```
Layout Managers

• Java’s layout managers provide a level of abstraction to automatically map your user interface on all window systems.

• The UI components are placed in containers. Each container has a layout manager to arrange the UI components within the container.

• Layout managers are set in containers using the `setLayout(LayoutManager)` method in a container.
Kinds of Layout Managers

• FlowLayout

• GridLayout

• BorderLayout

• Several other layout managers are available...
FlowLayout Example

- The FlowLayout class puts components in a row, sized at their preferred size.
- If the horizontal space in the container is too small to put all the components in one row, the FlowLayout class uses multiple rows.

```java
JPanel content = new JPanel();
content.setLayout(new FlowLayout());
content.add(new JButton("Button 1"));
content.add(new JButton("2"));
content.add(new JButton("This is button three"));
content.add(new JButton("four"));
```
### The FlowLayout Class

<table>
<thead>
<tr>
<th>java.awt.FlowLayout</th>
<th>The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-alignment: int</td>
<td>The alignment of this layout manager (default: CENTER).</td>
</tr>
<tr>
<td>-hgap: int</td>
<td>The horizontal gap of this layout manager (default: 5 pixels).</td>
</tr>
<tr>
<td>-vgap: int</td>
<td>The vertical gap of this layout manager (default: 5 pixels).</td>
</tr>
<tr>
<td>+FlowLayout()</td>
<td>Creates a default FlowLayout manager.</td>
</tr>
<tr>
<td>+FlowLayout(alignment: int)</td>
<td>Creates a FlowLayout manager with a specified alignment.</td>
</tr>
</tbody>
</table>
GridLayout Example

Lays out a container's components in a rectangular grid.

The container is divided into equal-sized rectangles, and one component is placed in each rectangle.

The following example uses a GridLayout that lays out 3 buttons and an empty label into two rows and two columns:

```java
JPanel content = new JPanel(new GridLayout(2, 2));
content.add(new JButton("Button 1"));
content.add(new JButton("2"));
content.add(new JLabel("")); // for empty cell
content.add(new JButton("This is button three"));
```
The BorderLayout Manager

The `BorderLayout` manager divides the container into five areas: East, South, West, North, and Center. Components are added to a `BorderLayout` by using the `add` method:

```
add(Component, constraint),
where constraint is
BorderLayout.EAST,
BorderLayout.SOUTH,
BorderLayout.WEST,
BorderLayout.NORTH, or
BorderLayout.CENTER.
```
# The BorderLayout Class

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

<table>
<thead>
<tr>
<th>java.awt.BorderLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>-hgap: int</td>
</tr>
<tr>
<td>-vgap: int</td>
</tr>
<tr>
<td>+BorderLayout()</td>
</tr>
<tr>
<td>+BorderLayout(hgap: int, vgap: int)</td>
</tr>
</tbody>
</table>

The horizontal gap of this layout manager (default: 0).

The vertical gap of this layout manager (default: 0).

Creates a default BorderLayout manager.

Creates a BorderLayout manager with a specified number of horizontal gap, and vertical gap.
The **Color** Class

You can set colors for GUI components by using the `java.awt.Color` class.

Colors are made of red, green, and blue components, each of which is represented by a byte value that describes its intensity, ranging from 0 (darkest shade) to 255 (lightest shade). This is known as the *RGB model*.

\[
\text{Color } c = \text{new Color}(r, g, b);
\]

\(r, g, \text{ and } b\) specify a color by its red, green, and blue components.

Example:

\[
\text{Color } c = \text{new Color}(228, 100, 255);
\]
Setting Colors

Thirteen standard colors (black, blue, cyan, darkGray, gray, green, lightGray, magenta, orange, pink, red, white, yellow) are defined as constants in java.awt.Color.

You can use the following methods to set the component’s background and foreground colors:

setBackgroundColor(Color c)

setForeground(Color c)

Example:

jbt.setBackgroundColor(Color.yellow);

jbt.setForeground(Color.red);
Interface ActionListener

- The listener interface for receiving action events.
- The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's `addActionListener` method.
- When the action event occurs, that object's `actionPerformed` method is invoked.
How to Write an Action Listener

• To write an Action Listener, follow the steps given below:

  – Declare an event handler class and specify that the class either implements an ActionListener interface or extends a class that implements an ActionListener interface. For example:

    • public class MyClass implements ActionListener {

  – Register an instance of the event handler class as a listener on one or more components. For example:

    • someComponent.addActionListener(instanceOfMyClass);

  – Include code that implements the methods in listener interface. For example:

    • public void actionPerformed(ActionEvent e) {
      • ...
      • }