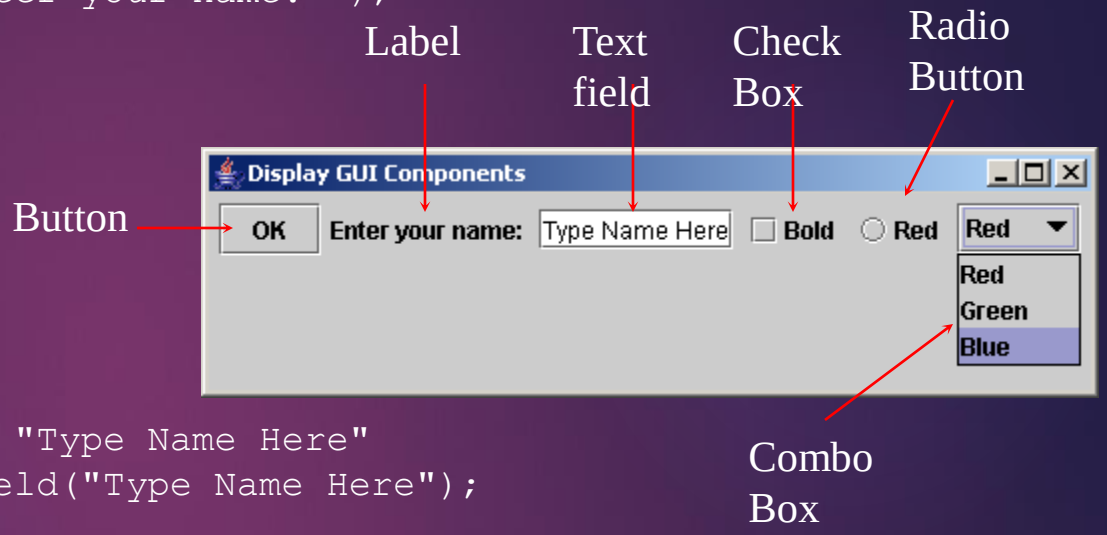
GUI Basics

The design of the API for Java GUI programming is an excellent example of how the object-oriented principle is applied. In the chapters that follow, you will learn the framework of Java GUI API and use the GUI components to develop user-friendly interfaces for applications and applets.

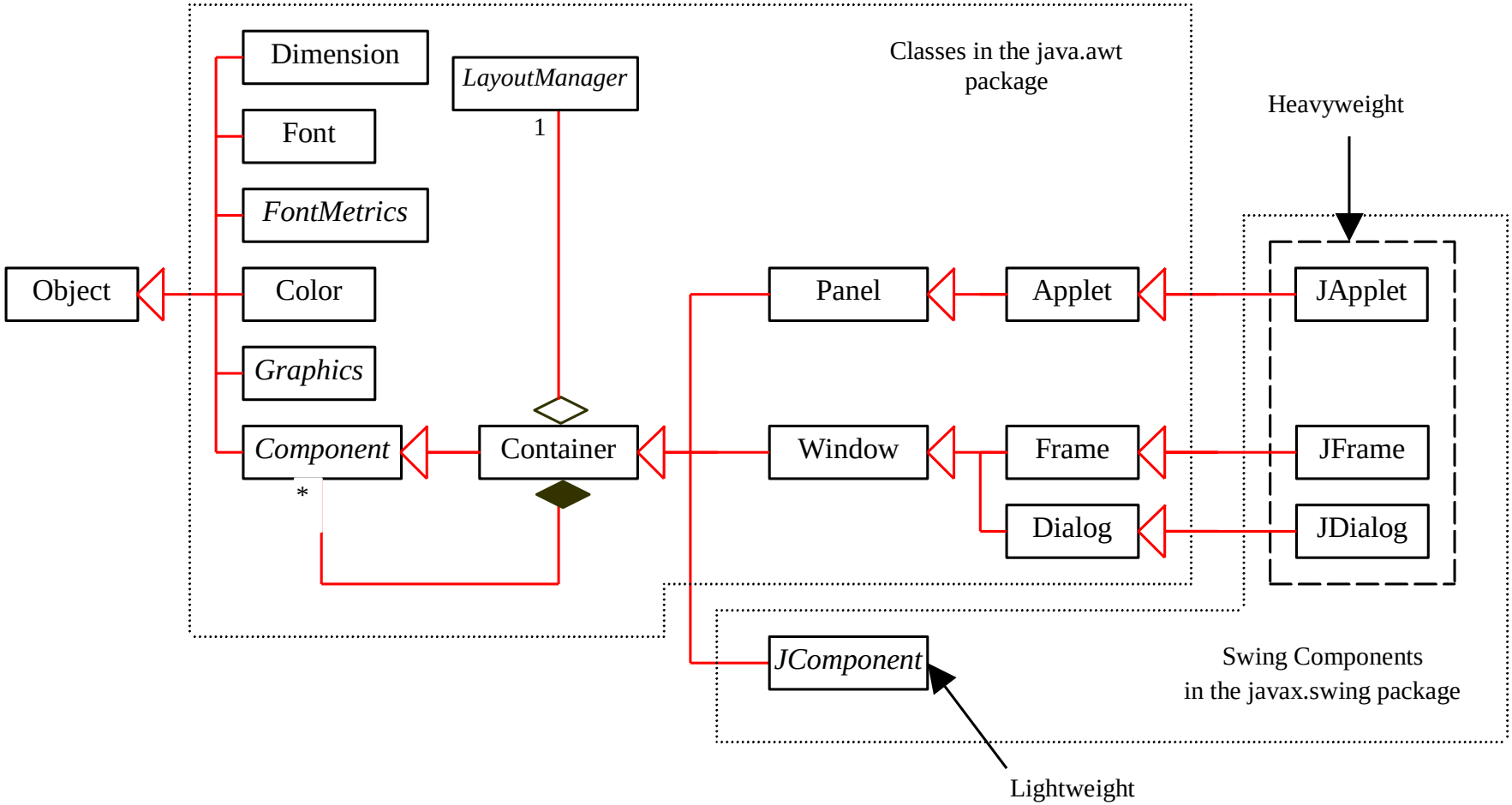
Creating GUI Objects


```
// 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: ");
```

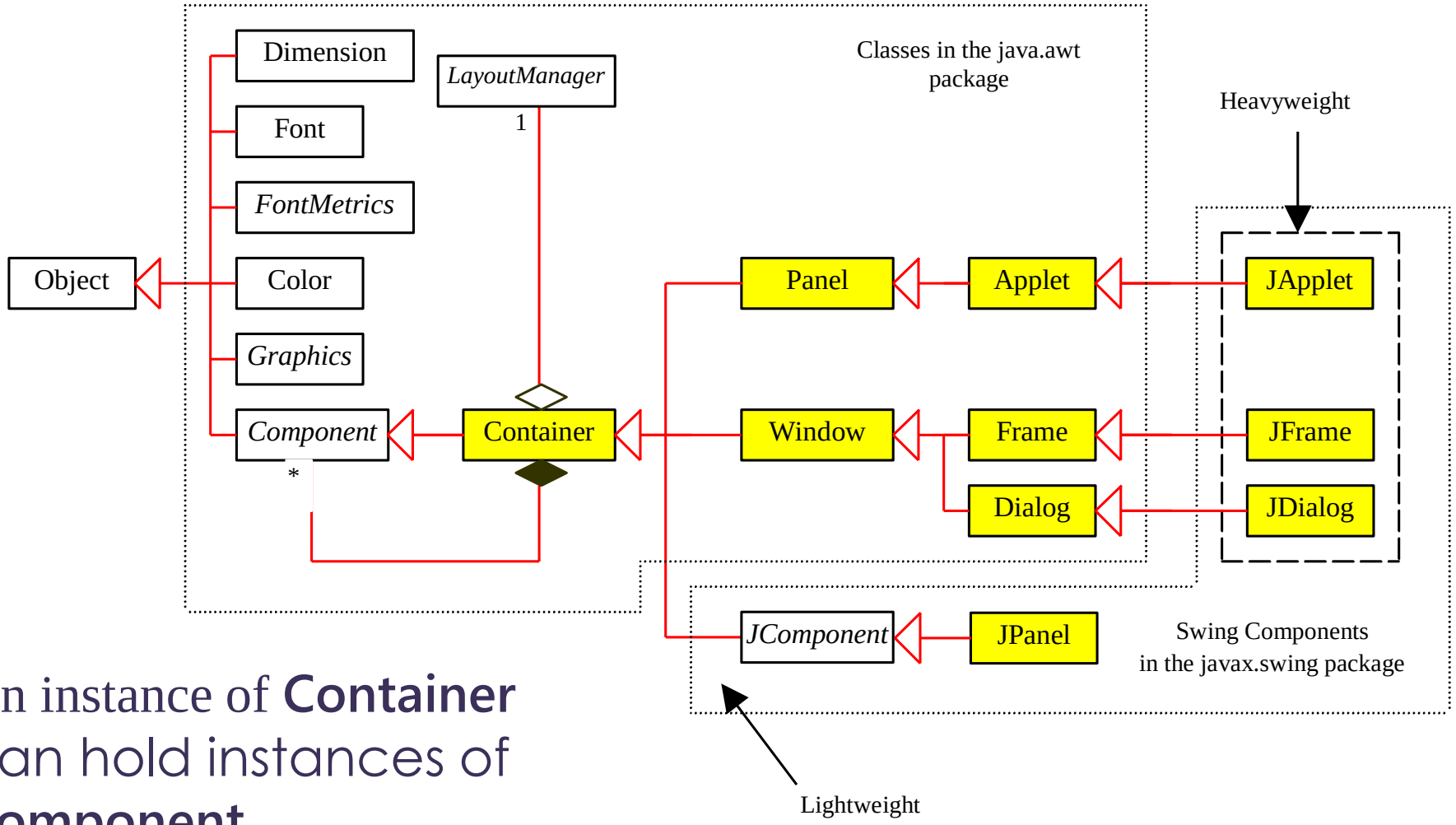



```
// Create a text field with text "Type Name Here"
JTextField jtfName = new JTextField("Type Name Here");

// 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"});
```

GUI Class Hierarchy (Swing)

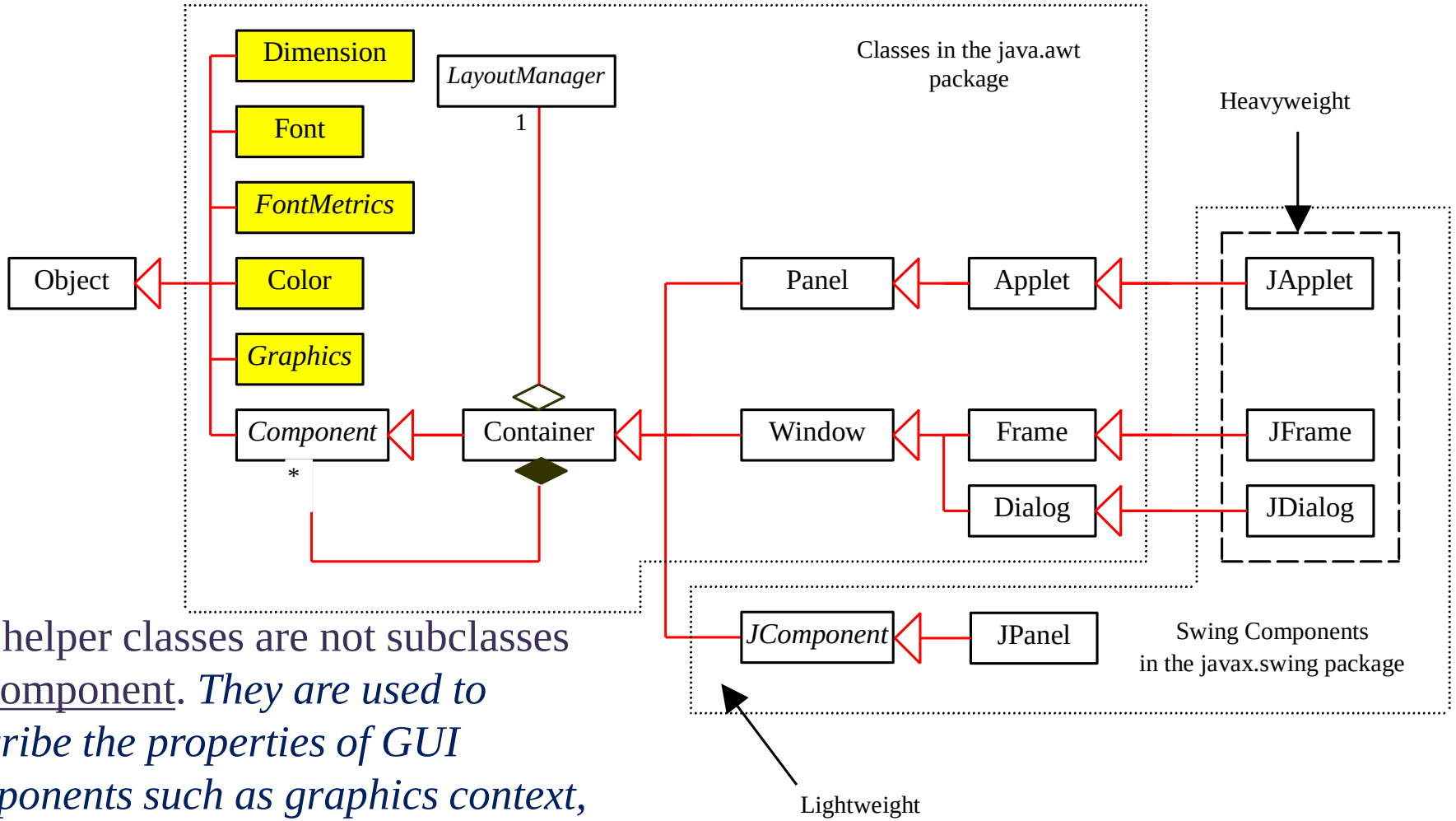


Container Classes



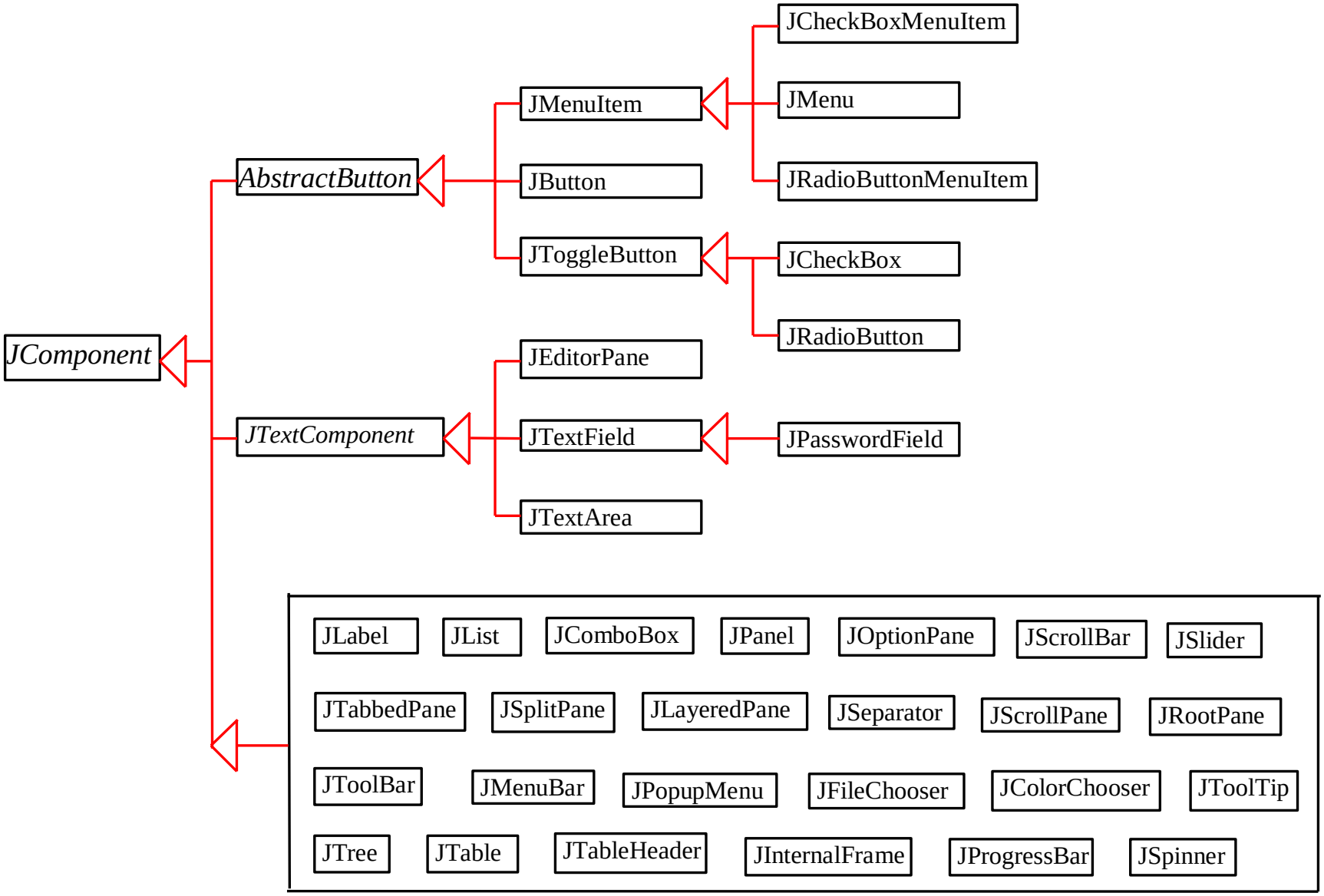
An instance of **Container** can hold instances of **Component**.

GUI Helper Classes

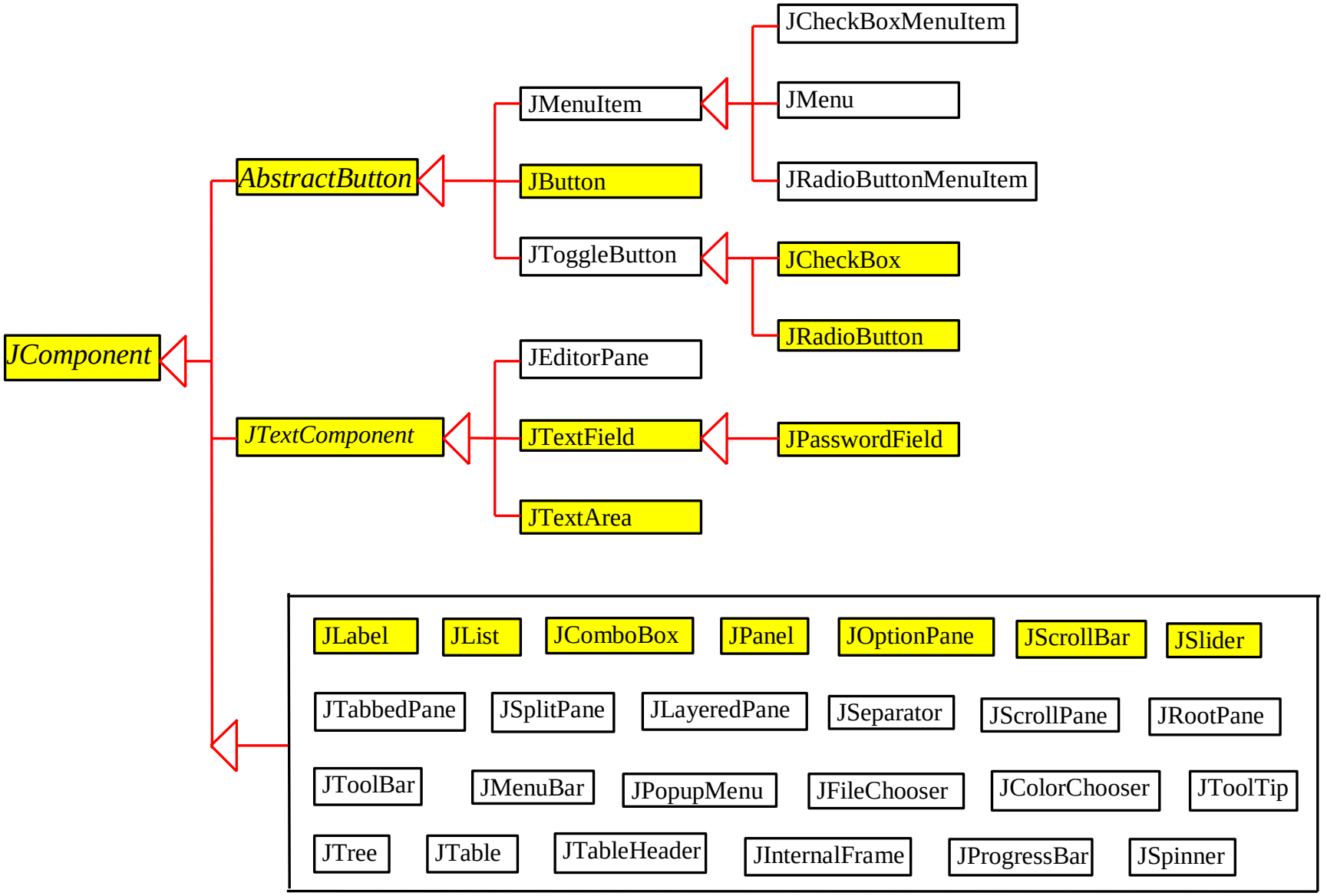


The helper classes are not subclasses of Component. *They are used to describe the properties of GUI components such as graphics context, colors, fonts, and dimension.*

Swing GUI Components



Components Covered in the Brief



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 widows.

Creating Frames


```
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);
  }
}
```

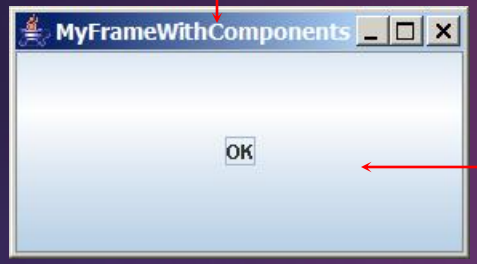
MyFrame

Run

Adding Components into a Frame


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

Title bar



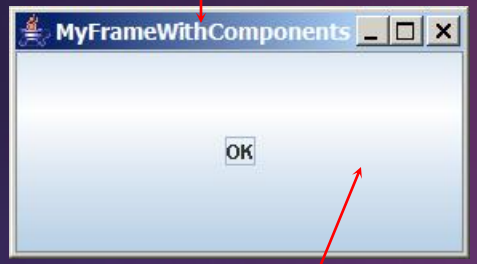
Content pane

MyFrameWithComponents

Run

Content Pane Delegation in JDK 1.5

Title bar



Content pane

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

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

JFrame Class

javax.swing.JFrame	
+JFrame()	Creates a default frame with no title.
+JFrame(title: String)	Creates a frame with the specified title.
+setSize(width: int, height: int): void	Specifies the size of the frame.
+setLocation(x: int, y: int): void	Specifies the upper-left corner location of the frame.
+setVisible(visible: boolean): void	Sets true to display the frame.
+setDefaultCloseOperation(mode: int): void	Specifies the operation when the frame is closed.
+setLocationRelativeTo(c: Component): void	Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.
+pack(): void	Automatically sets the frame size to hold the components in the frame.

Layout Managers

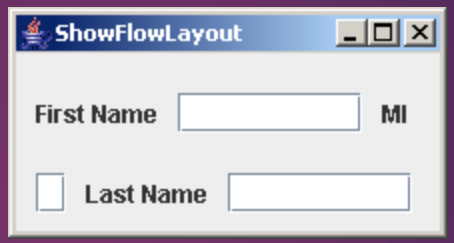
- Java's layout managers provide a level of abstraction to automatically map your user interface on all window systems.
- The UI components are placed in containers. Each container has a layout manager to arrange the UI components within the container.
- Layout managers are set in containers using the setLayout(LayoutManager) method in a container.

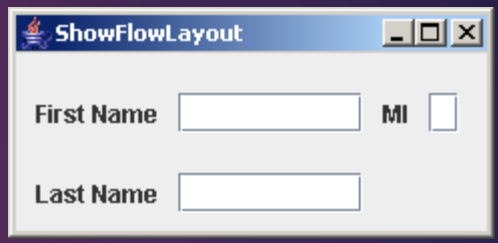
Kinds of Layout Managers

- FlowLayout
- GridLayout
- BorderLayout
- … and others

FlowLayout Example

Write a program that adds three labels and text fields into the content pane of a frame with a FlowLayout manager.

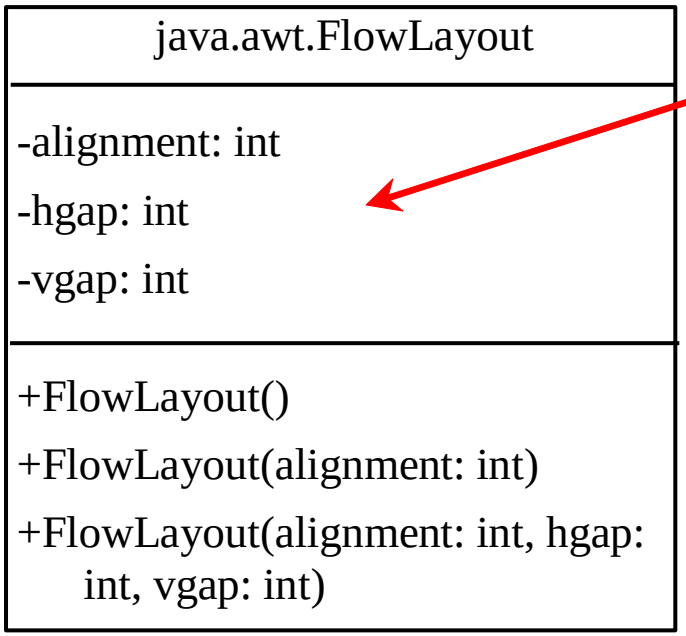




ShowFlowLayout

Run

The FlowLayout Class



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

The alignment of this layout manager (default: CENTER).

The horizontal gap of this layout manager (default: 5 pixels).

The vertical gap of this layout manager (default: 5 pixels).

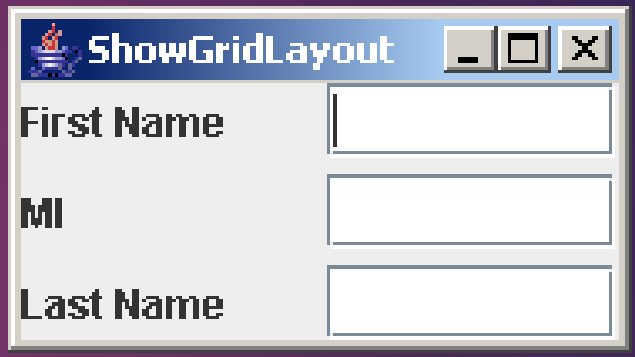
Creates a default FlowLayout manager.

Creates a FlowLayout manager with a specified alignment.

Creates a FlowLayout manager with a specified alignment, horizontal gap, and vertical gap.

GridLayout Example

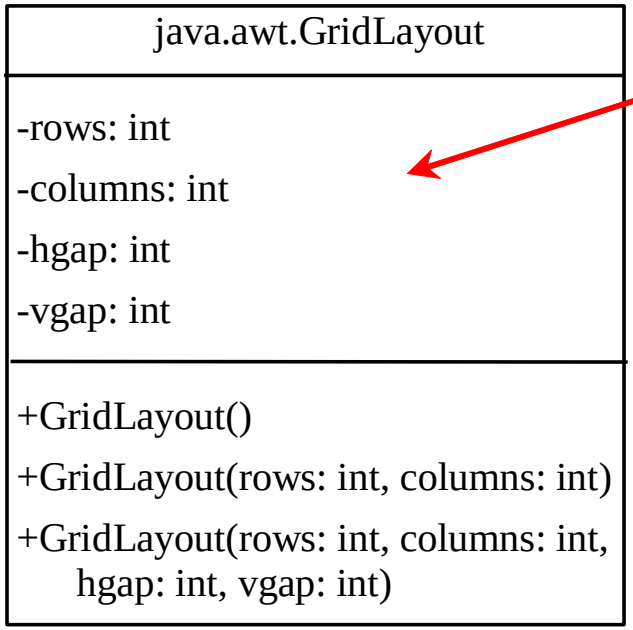
Rewrite the program in the preceding example using a GridLayout manager instead of a FlowLayout manager to display the labels and text fields.



ShowGridLayout

Run

The `GridLayout` Class



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

The number of rows in this layout manager (default: 1).

The number of columns in this layout manager (default: 1).

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

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

Creates a default GridLayout manager.

Creates a GridLayout with a specified number of rows and columns.

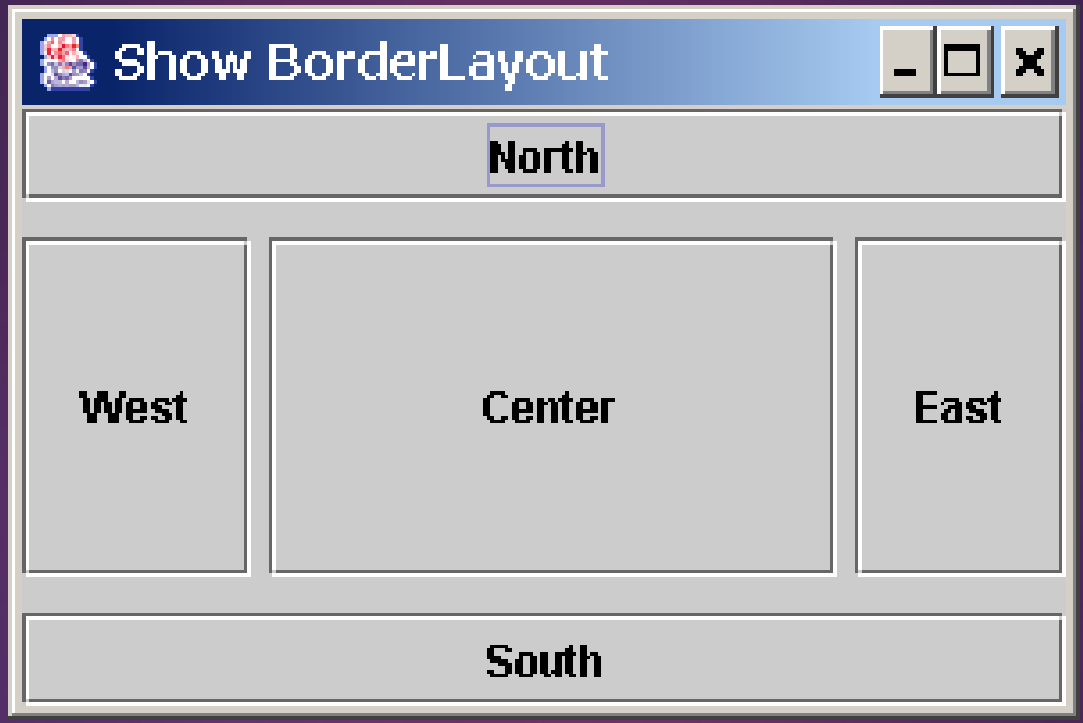
Creates a GridLayout manager with a specified number of rows and columns, horizontal gap, and vertical gap.

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`.

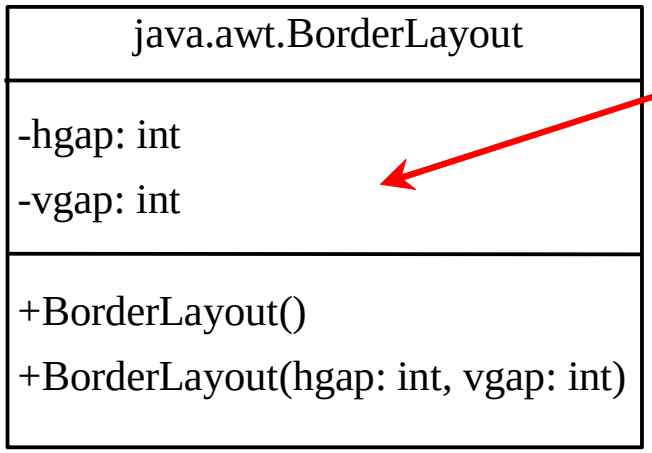
BorderLayout Example



ShowBorderLayout

Run

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.

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*.

```
Color c = new Color(r, g, b);
```

`r`, `g`, and `b` specify a color by its red, green, and blue components.

Example:

```
Color c = new Color(228, 100, 255);
```

Standard 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.

Setting Colors

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

```
setBackground(Color c)
setForeground(Color c)
```

Example:

```
jbt.setBackground(Color.yellow);
jbt.setForeground(Color.red);
```

The Font Class

Font Names

Standard font names that are supported in all platforms are: SansSerif, Serif, Monospaced, Dialog, or DialogInput.

Font Style

Font.PLAIN (0), Font.BOLD (1), Font.ITALIC (2), and Font.BOLD + Font.ITALIC (3)

```
Font myFont = new Font(name, style, size);
```

Example:

```
Font myFont = new Font("SansSerif ", Font.BOLD, 16);
Font myFont = new Font("Serif", Font.BOLD+Font.ITALIC, 12);

JButton jbtOK = new JButton("OK");
jbtOK.setFont(myFont);
```

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.

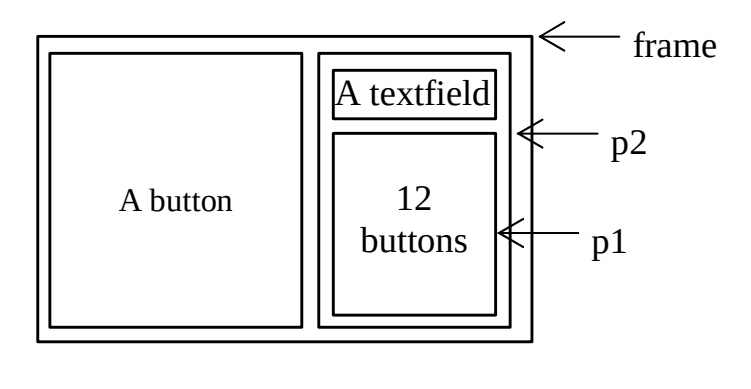
Creating a JPanel

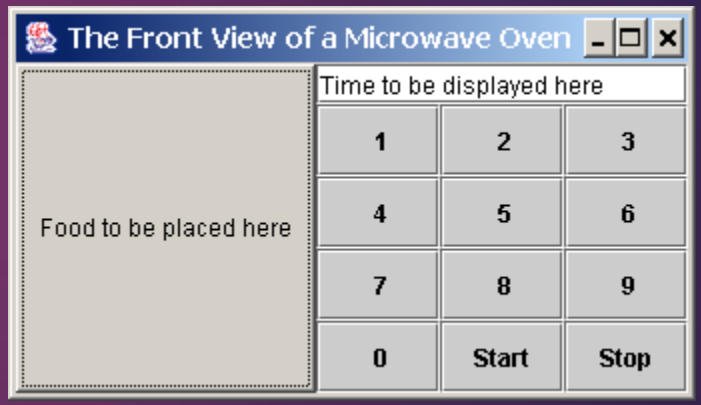
You can use new JPanel() to create a panel with a default FlowLayout manager or new JPanel(LayoutManager) to create a panel with the specified layout manager. Use the add(Component) method to add a component to the panel. For example,

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

Testing Panels Example

This example uses panels to organize components. The program creates a user interface for a Microwave oven.

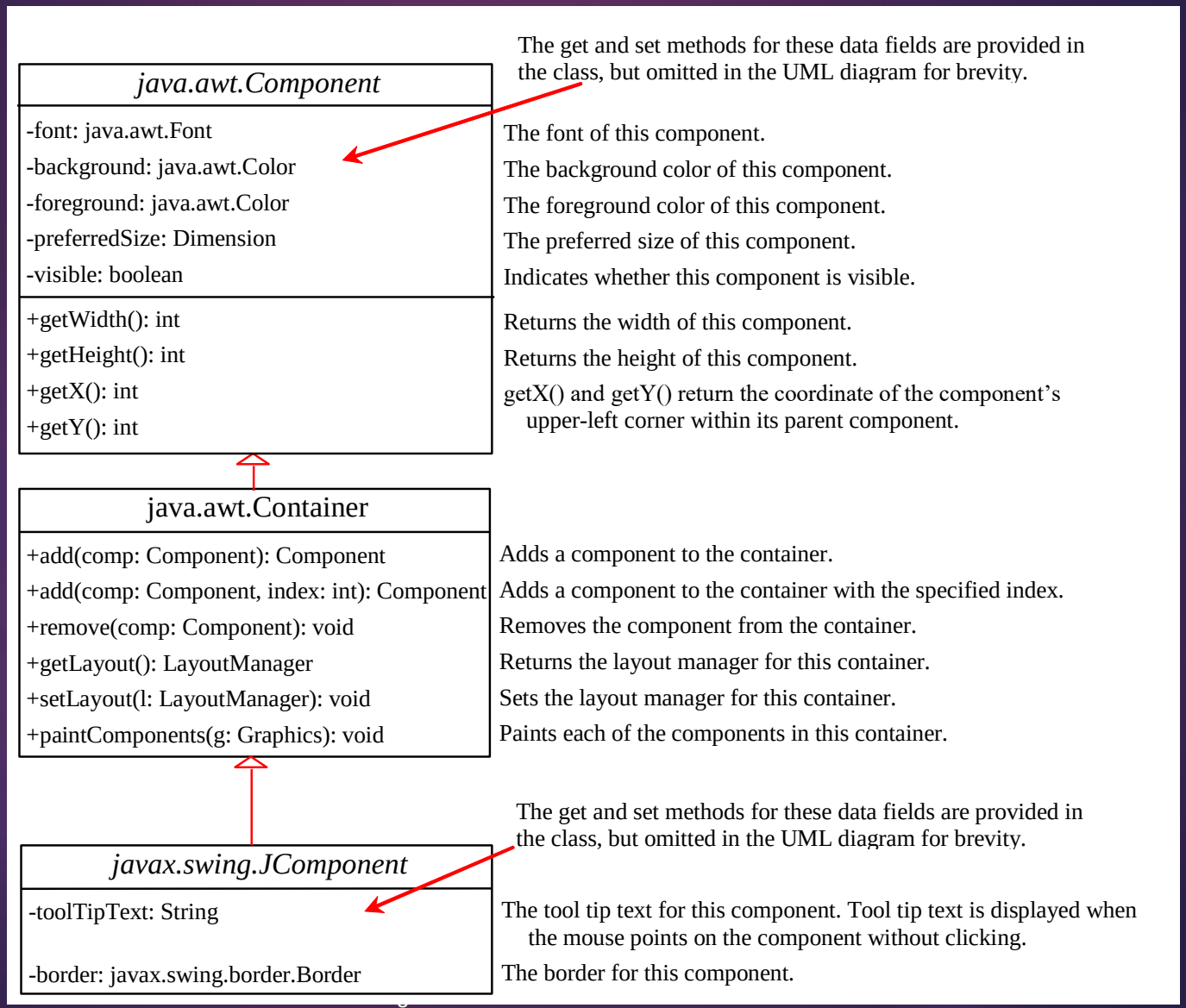




TestPanels

Run

Common Features of Swing Components



Borders

You can set a border on any object of the JComponent class. Swing has several types of borders. To create a titled border, use

new TitledBorder(String title).

To create a line border, use

new LineBorder(Color color, int width),

where width specifies the thickness of the line. For example, the following code displays a titled border on a panel:

```
JPanel panel = new JPanel();
panel.setBorder(new TitleBorder("My Panel"));
```