Module 12

Building Java GUIs Using the Swing API

Objectives

- Describe the JFC Swing technology
- Define Swing
- Identify the Swing packages
- Describe the GUI building blocks: containers, components, and layout managers
- Examine top-level, general-purpose, and special-purpose properties of container
- Examine components
- Examine layout managers
- Describe the Swing single-threaded model
- Build a GUI using Swing components

What Are the Java Foundation Classes (JFC)?

Java Foundation Classes are a set of Graphical User Interface (GUI) support packages, including:

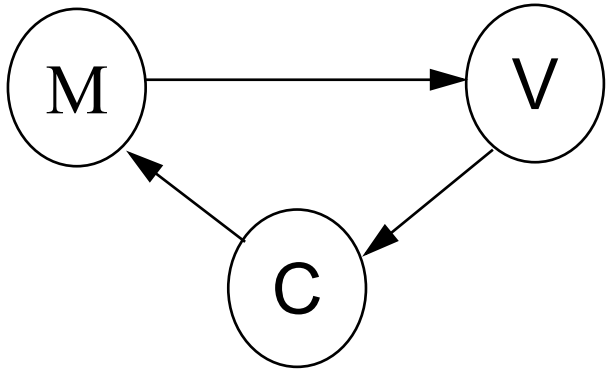
- Abstract Window Toolkit (AWT)
- The Swing component set
- 2D graphics
- Pluggable look-and-feel
- Accessibility
- Drag-and-drop
- Internationalization

What Is Swing?

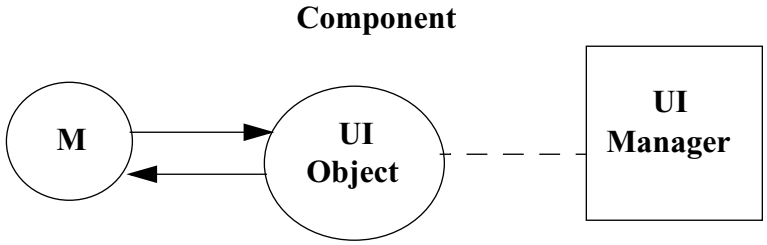
- An enhanced GUI component set
- Provides replacement components for those in the original AWT
- Has special features, such as a pluggable look-and feel

Swing Architecture

- Has its roots in the Model-View-Controller (MVC) architecture



- The Swing components follow Separable Model Architecture



Swing Packages

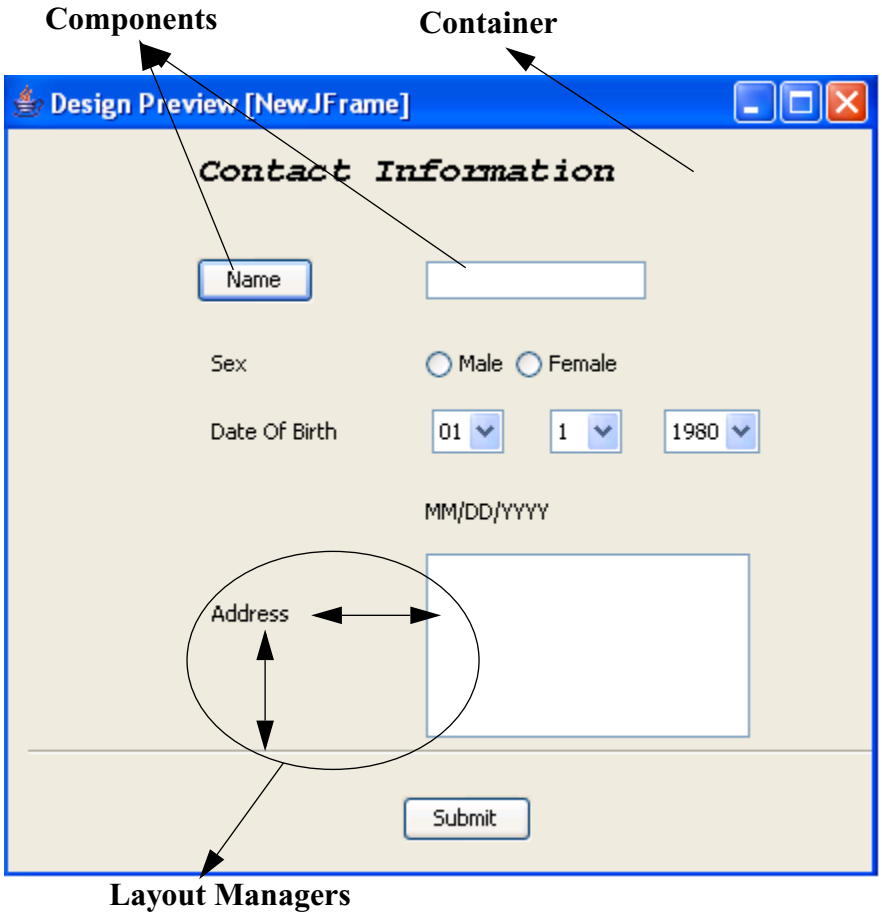
Package Name	Package Name
javax.swing	javax.swing.colorchooser
javax.swing.border	javax.swing.filechooser
javax.swing.event	javax.swing.table
javax.swing.undo	javax.swing.tree
javax.swing.plaf	javax.swing.text
javax.swing.plaf.basic	javax.swing.text.html
javax.swing.plaf.metal	javax.swing.text.html.parser
javax.swing.plaf.multi	javax.swing.text.rtf
javax.swing.plaf.synth	javax.swing.undo

Examining the Composition of a Java Technology GUI

A Swing API-based GUI is composed of the following elements:

- Containers – Are on top of the GUI containment hierarchy.
- Components – Contain all the GUI components that are derived from the `JComponent` class.
- Layout Managers – Are responsible for laying out components in a container.

Examining the Composition of a Java Technology GUI

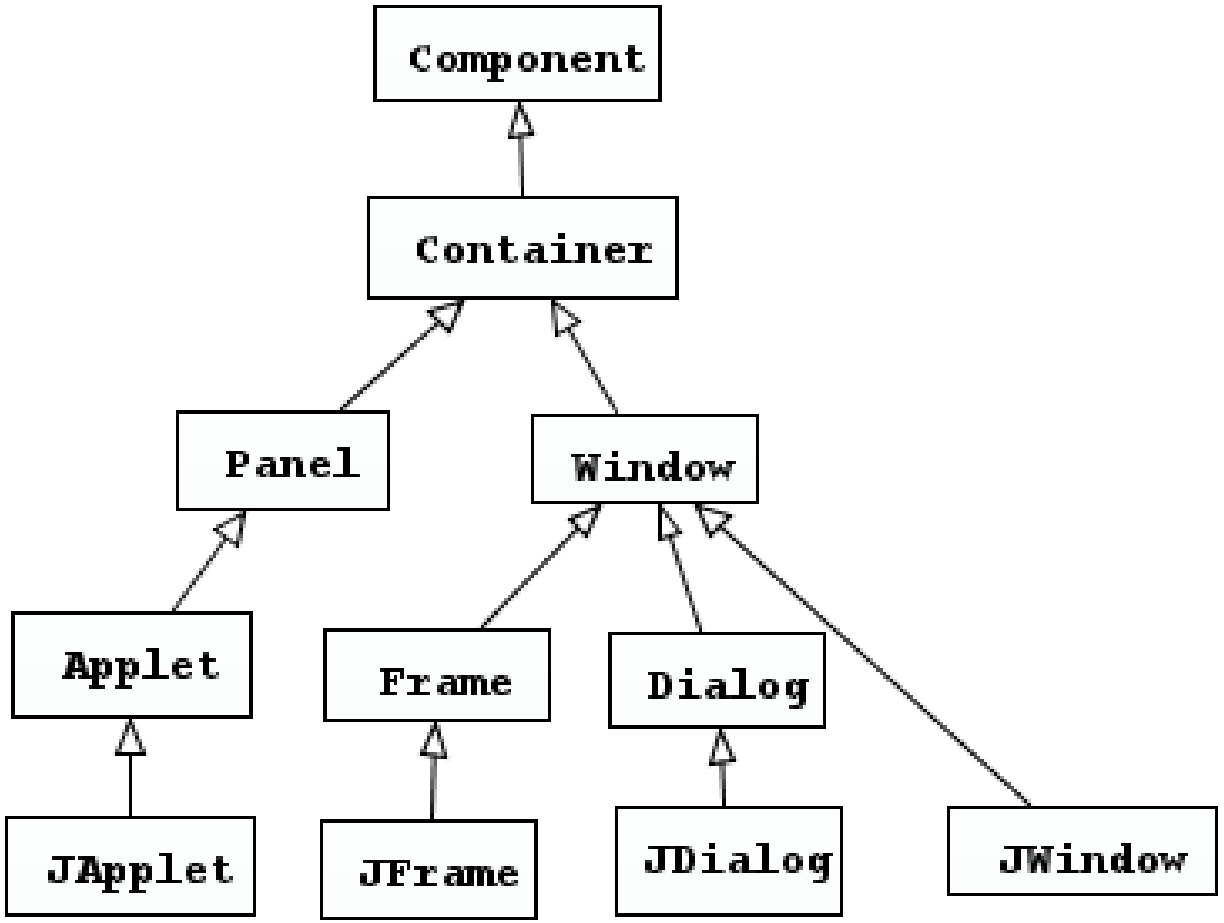


Swing Containers

Swing containers can be classified into three main categories:

- Top-level containers:
  - `JFrame`, `JWindow`, and `JDialog`
- General-purpose containers:
  - `JPanel`, `JScrollPane`,`JToolBar`,`JSplitPane`, and `JTabbedPane`
- Special-purpose containers:
  - `JInternalFrame` and `JLayeredPane`

Top-Level Containers

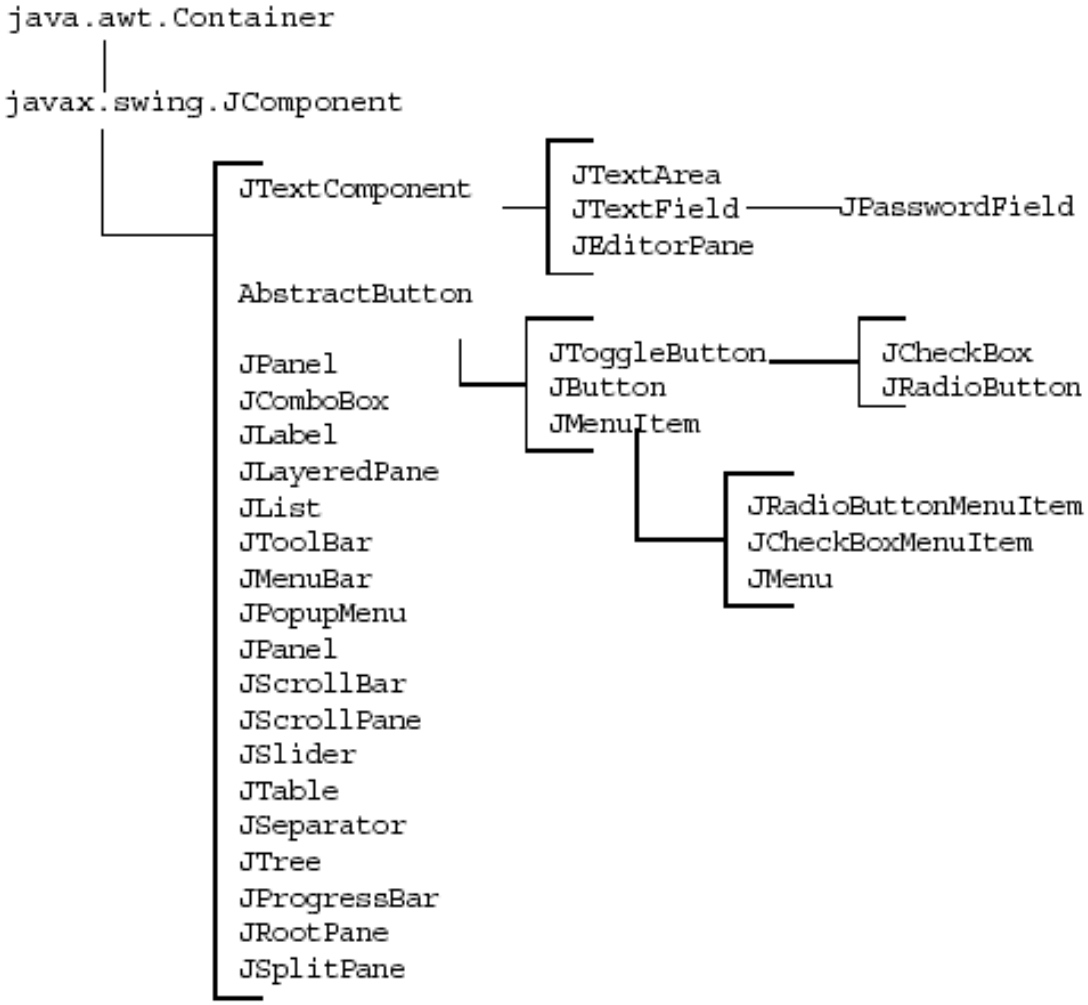


Swing Components

Swing components can be broadly classified as:

- Buttons
- Text components
- Uneditable information display components
- Menus
- Formatted display components
- Other basic controls

Swing Component Hierarchy



Text Components

Swing text components can be broadly divided into three categories.

- Text controls – `JTextField`, `JPasswordField` (for user input)
- Plain text areas – `JTextArea` (displays text in plain text, also for multi-line user input
- Styled text areas – `JEditorPane`, `JTextPane` (displays formatted text)

Swing Component Properties

Common component properties:

- All the Swing components share some common properties because they all extend `JComponent`.

Component-specific properties:

- Each component defines more specific properties.

Common Component Properties

Property	Methods
Border	`Border getBorder()` `void setBorder(Border b)`
Background and foreground color	`void setBackground(Color bg)` `void setForeground(Color bg)`
Font	`void setFont(Font f)`
Opaque	`void setOpaque(boolean isOpaque)`
Maximum and minimum size	`void setMaximumSize(Dimension d)` `void setMinimumSize(Dimension d)`
Alignment	`void setAlignmentX(float ax)` `void setAlignmentY(float ay)`
Preferred size	`void setPreferredSize(Dimension ps)`

Component-Specific Properties

The following shows properties specific to `JComboBox`.

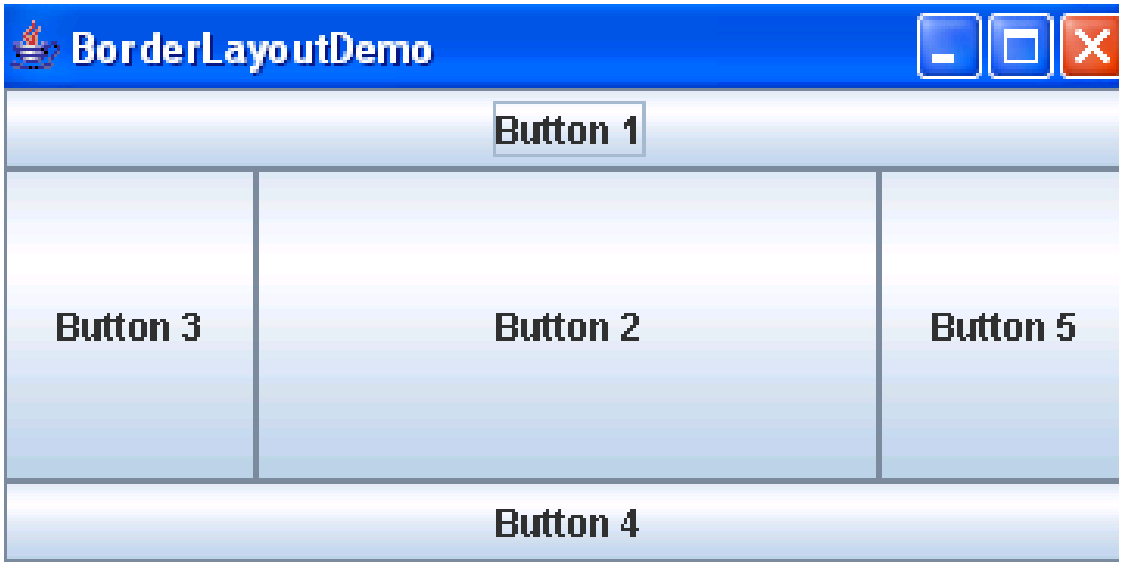
Properties	Methods
Maximum row count	`void setMaximumRowCount(int count)`
Model	`void setModal(ComboBoxModel cbm)`
Selected index	`int getSelectedIndex()`
Selected Item	`Object getSelectedItem()`
Item count	`int getItemCount()`
Renderer	`void setRenderer(ListCellRenderer ar)`
Editable	`void setEditable(boolean flag)`

Layout Managers

- Handle problems caused by:
  - GUI resizing by user
  - Operating system differences in fonts
  - Locale-specific text layout requirements
- Layout manager classes:
  - `BorderLayout`
  - `FlowLayout`
  - `BoxLayout`
  - `CardLayout`
  - `GridLayout`
  - `GridBagLayout`

The `BorderLayout` Manager

The `BorderLayout` manager places components in top, bottom, left, right and center locations.



BorderLayout Example

```
1   import java.awt.*;
2   import javax.swing.*;
3
4   public class BorderExample {
5     private JFrame f;
6     private JButton bn, bs, bw, be, bc;
7
8     public BorderExample() {
9       f = new JFrame("Border Layout");
10      bn = new JButton("Button 1");
11      bc = new JButton("Button 2");
12      bw = new JButton("Button 3");
13      bs = new JButton("Button 4");
14      be = new JButton("Button 5");
15    }
16
```

BorderLayout Example

```
17    public void launchFrame() {
18      f.add(bn, BorderLayout.NORTH);
19      f.add(bs, BorderLayout.SOUTH);
20      f.add(bw, BorderLayout.WEST);
21      f.add(be, BorderLayout.EAST);
22      f.add(bc, BorderLayout.CENTER);
23      f.setSize(400,200);
24      f.setVisible(true);
25    }
26
27    public static void main(String args[]) {
28      BorderExample guiWindow2 = new BorderExample();
29      guiWindow2.launchFrame();
30    }
31  }
32
```

The `FlowLayout` Manager

The `FlowLayout` manager places components in a row, and if the row fills, components are placed in the next row.

FlowLayout Example

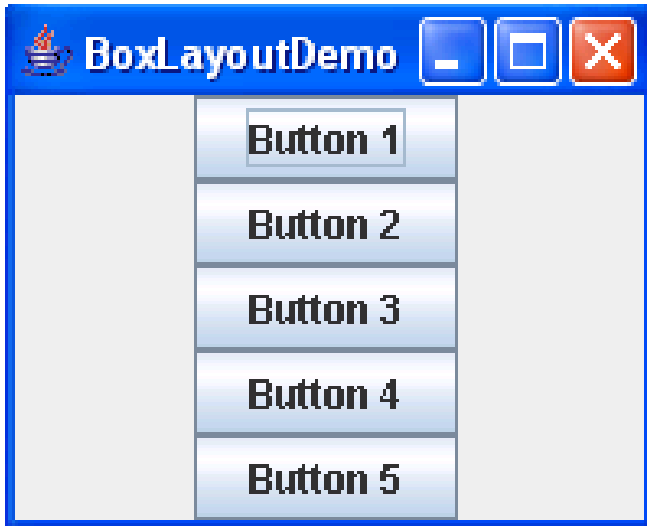
```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class LayoutExample {
5       private JFrame f;
6       private JButton b1;
7       private JButton b2;
8       private JButton b3;
9       private JButton b4;
10      private JButton b5;
11
12      public LayoutExample() {
13          f = new JFrame("GUI example");
14          b1 = new JButton("Button 1");
15          b2 = new JButton("Button 2");
16          b3 = new JButton("Button 3");
17          b4 = new JButton("Button 4");
18          b5 = new JButton("Button 5");
19      }
```

FlowLayout Example

```
20
21      public void launchFrame() {
22          f.setLayout(new FlowLayout());
23          f.add(b1);
24          f.add(b2);
25          f.add(b3);
26          f.add(b4);
27          f.add(b5);
28          f.pack();
29          f.setVisible(true);
30      }
31
32      public static void main(String args[]) {
33          LayoutExample guiWindow = new LayoutExample();
34          guiWindow.launchFrame();
35      }
36
37  } // end of LayoutExample class
```

The BoxLayout Manager

The BoxLayout manager adds components from left to right, and from top to bottom in a single row of column.

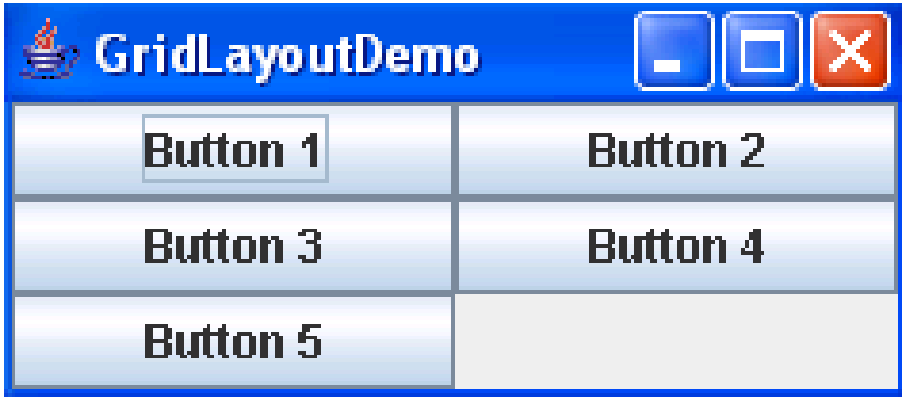


The `CardLayout` Manager

The `CardLayout` manager places the components in different cards. Cards are usually controlled by a combo box.

The `GridLayout` Manager

The `GridLayout` manager places components in rows and columns in the form of a grid.



GridLayout Example

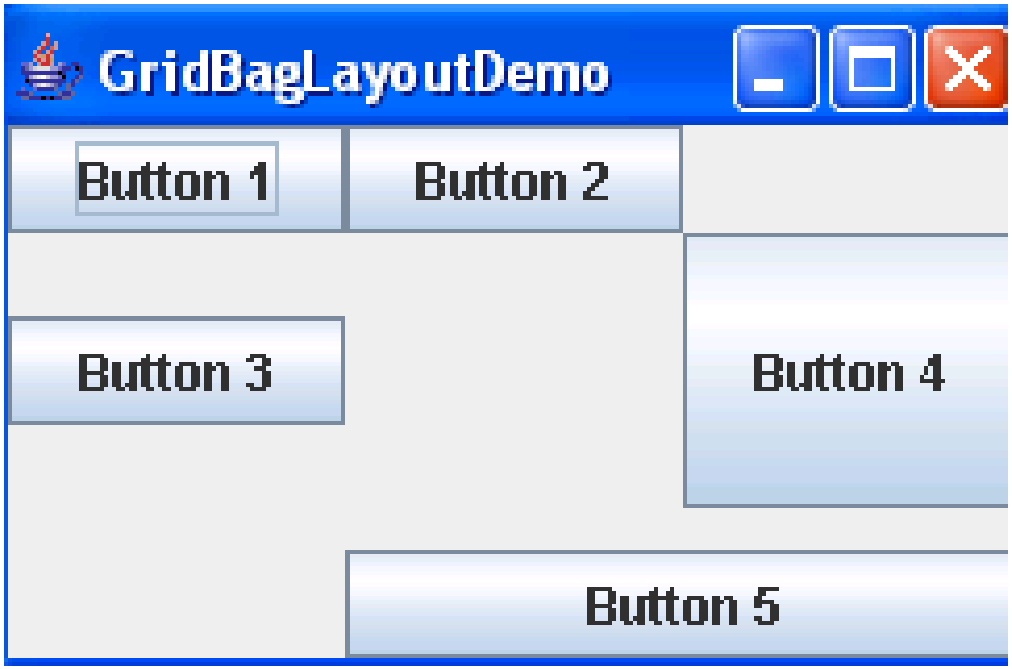
```
1   import java.awt.*;
2   import javax.swing.*;
3
4   public class GridExample {
5     private JFrame f;
6     private JButton b1, b2, b3, b4, b5;
7
8     public GridExample() {
9       f = new JFrame("Grid Example");
10      b1 = new JButton("Button 1");
11      b2 = new JButton("Button 2");
12      b3 = new JButton("Button 3");
13      b4 = new JButton("Button 4");
14      b5 = new JButton("Button 5");
15    }
16
```

GridLayout Example

```
17  public void launchFrame() {
18      f.setLayout (new GridLayout(3,2));
19
20      f.add(b1);
21      f.add(b2);
22      f.add(b3);
23      f.add(b4);
24      f.add(b5);
25
26      f.pack();
27      f.setVisible(true);
28    }
29
30    public static void main(String args[]) {
31      GridExample grid = new GridExample();
32      grid.launchFrame();
33    }
34  }
```

The `GridBagLayout` Manager

The `GridBagLayout` manager arranges components in rows and columns, similar to a grid layout, but provides a wide variety of options for resizing and positioning the components.



GUI Construction

- Programmatic
- GUI builder tool

Programmatic Construction

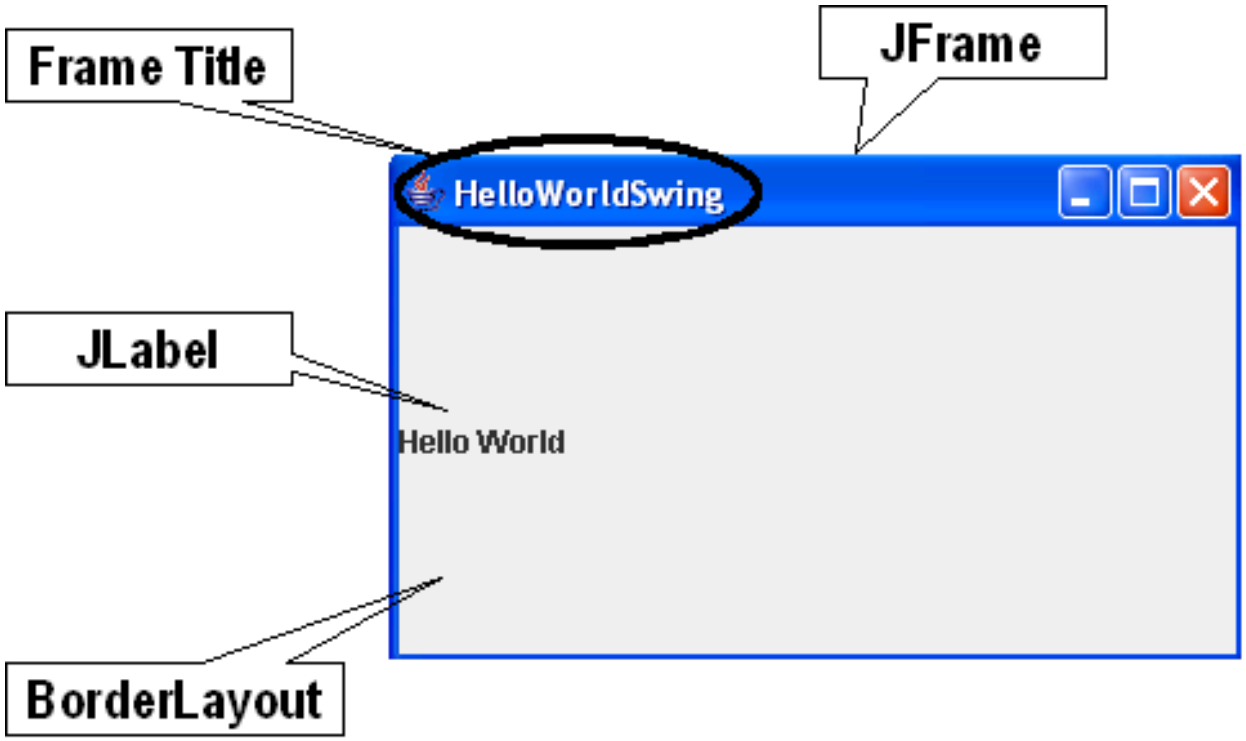
```
1   import javax.swing.*;
2   public class HelloWorldSwing {
3       private static void createAndShowGUI() {
4           JFrame frame = new JFrame("HelloWorldSwing");
5         //Set up the window.
6           frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7           JLabel label = new JLabel("Hello World");
8         // Add Label
9           frame.add(label);
10          frame.setSize(300,200);
11        // Display Window
12          frame.setVisible(true);}
13
```

Programmatic Construction

```
14  public static void main(String[] args) {
15          javax.swing.SwingUtilities.invokeLater(new Runnable() {
16              //Schedule for the event-dispatching thread:
17              //creating,showing this app's GUI.
18             public void run() {createAndShowGUI();}
19          });
20      }
21  }
```

Programmatic Construction

The output generated from the program



Key Methods

Methods for setting up the `JFrame` and adding `JLabel`:

- `setDefaultCloseOperationJFrame.EXIT_ON_CLOSE)` –Creates the program to exit when the close button is clicked.
- `setVisible(true)`– Makes the `JFrame` visible.
- `add(label)`– `JLabel` is added to the content pane not to the `JFrame` directly.

Key Methods

- Tasks:
  - Executing GUI application code, such as rendering
  - Handling GUI events
  - Handling time consuming (background) processes
- The `SwingUtilities` class:
  - `SwingUtilites.invokeLater(new Runnable())`