CS 251
Intermediate Programming
GUIs: Components and Layout

Brooke Chenoweth

University of New Mexico

Spring 2024

Hello GUI

```
import javax.swing.*;

public class HelloGUI extends JFrame {
  public HelloGUI() {
    super("Hello World GUI");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    getContentPane().add(new JLabel("Hello, World"));
    pack();
  }

  public static void main(String[] args) {
    new HelloGUI().setVisible(true);
  }
}
```

Java Packages

We'll be using two major packages when writing our GUIs

- `javax.swing`
  - Provides most of the GUI components.
- `java.awt`
  - Provides graphics, fonts, layouts.

GUI components and containers

- `java.awt.Component` Abstract class for graphics objects.
- `java.awt.Container` Component that can contain other components.
- `javax.swing.JComponent` Base class for all Swing components (except top-level containers)

Top-Level Containers

- `JFrame` – top-level window
- `JDialog` – temporary subwindow
- `JApplet` – embed in web page, run in browser

Top-level container is the root of a *containment hierarchy*.

The *content pane* holds the components.

JFrame

- Top-level window with title and border.
- For main window of application, usually want to change default close operation from hide to exit.

```
JFrame frame = new JFrame("My Frame Title");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JDialog

- Temporary subwindow for messages, warnings, user input, etc.
- Display relative to a *parent component* (usually the main frame of the application)
- Often *modal*, blocking input to other windows in the program.
- `JOptionPane` provides methods to create many standard dialogs.
- `JColorChooser` and `JFileChooser` also useful.
- Use `JDialog` directly for custom work.

Swing Components

- Buttons
- Check Boxes
- Labels
- Panels
- Text Areas
- Tables
- etc. . . .

Use the Java Tutorials and API Reference.

JComponent methods

- Appearance
- State
- Events
- Painting
- Containment Hierarchy
- Layout
- Size and Position

JComponent methods

- Appearance
  - Borders
  - Foreground and background colors
  - Opaque
  - Font
  - Cursor
- State
- Events
- Painting
- Containment Hierarchy
- Layout
- Size and Position

JComponent methods

- Appearance
- State
  - Name
  - Tool Tip
  - Visible
  - Enabled
- Events
- Painting
- Containment Hierarchy
- Layout
- Size and Position

JComponent methods

- Appearance
- State
- Events
  - Keyboard
  - Mouse moves
  - Mouse clicks
- Painting
- Containment Hierarchy
- Layout
- Size and Position

JComponent methods

- Appearance
- State
- Events
- Painting
  - Force repaint
  - Custom paint by overriding `paintComponent`
- Containment Hierarchy
- Layout
- Size and Position

JComponent methods

- Appearance
- State
- Events
- Painting
- Containment Hierarchy
  - Add/remove components
  - Get parent component
  - Get contained components
- Layout
- Size and Position

JComponent methods

- Appearance
- State
- Events
- Painting
- Containment Hierarchy
- Layout
  - Preferred, Maximum, Minimum Size
  - `LayoutManager`
- Size and Position

JComponent methods

- Appearance
- State
- Events
- Painting
- Containment Hierarchy
- Layout
- Size and Position
  - Size in pixels
  - Location

Layout Managers

- FlowLayout
- BorderLayout
- BoxLayout
- GridLayout
- GridBagLayout

FlowLayout

- Components in a row, wrapping as needed.
- Default for `JPanel`
- Row is centered by default.

FlowLayout Example

```
import javax.swing.*;

public class FlowLayoutDemo {
  public static void main(String[] args) {
    JFrame frame = new JFrame("FlowLayout Demo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel(); // Defaults to FlowLayout
    panel.add(new JButton("One"));
    panel.add(new JButton("Two"));
    panel.add(new JButton("Three"));
    panel.add(new JButton("Four"));
    panel.add(new JButton("Five"));

    frame.setContentPane(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```

BorderLayout

- Add components to center or border areas.
  - PAGE_START
  - PAGE_END
  - LINE_START
  - LINE_END
  - CENTER
- Default for `JFrame`'s content pane.
- Center area gets the most space.
- Don't need to use all areas.

BorderLayout Example

```
import javax.swing.*;
import java.awt.BorderLayout;

public class BorderLayoutDemo {
  public static void main(String[] args) {
    JFrame frame = new JFrame("BorderLayout Demo");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel(new BorderLayout());
    panel.add(new JButton("Page Start"), BorderLayout.PAGE_START);
    panel.add(new JButton("Page End"), BorderLayout.PAGE_END);
    panel.add(new JButton("Line Start"), BorderLayout.LINE_START);
    panel.add(new JButton("Line End"), BorderLayout.LINE_END);
    panel.add(new JButton("Center"), BorderLayout.CENTER);

    frame.setContentPane(panel);
    frame.pack();
    frame.setVisible(true);
  }
}
```

BoxLayout

- Stacks components vertically or places in horizontal row – your choice.
  - LINE_AXIS
  - PAGE_AXIS
- Like `FlowLayout` with more functionality.
- Use invisible components to adjust spacing.
  - Defined in `Box` class
  - Rigid area – fixed space between components
  - Glue – Specify where excess space should go
  - Filler – component with min, max, preferred size.

Panels

- A `JPanel` is a lightweight container to hold other components (including other panels)
- Default layout is `FlowLayout`
- Use to logically group components
- Panels in panels can provide powerful layout control.

Event Listeners

- Look at component to see what sort of listeners it can take.
- Implement listener interface and add to component.
- Listener interfaces with multiple methods generally have an *adapter* class that provides empty implementation.