Lecture 18
Java User Interface Programming

• **AWT**
  – Original Java UI framework
  – Normalized interface to native OS UI toolkits
  – Advantages and Disadvantages
  – Packages start with `java.awt`

• **Swing**
  – Pure Java
  – More sophisticated components
  – More easily extended and customized
  – Packages start with `javax.swing`
    • Class names generally start with “J”
Hello, World

• lec18.v1
Hello, World

JFrame main_frame = new JFrame();
main_frame.setTitle("Hello, World");
main_frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

• JFrame
  – Top level window
  – setTitle() : text in window title
  – setDefaultCloseOperation()
    • What happens when window is closed.
Hello, World

JPanel main_panel = new JPanel();
main_frame.setContentPane(main_panel);

• JPanel
  – Generic container window.
    • Used to contain other user interface elements.

• main_frame.setContentPane(main_panel)
  – Top-level windows have a “content pane”
    • Main area of window.
    • Other areas of window: menu bar, window title, etc.
  – Replacing default content pane with our JPanel
    • Accept this as cookbook for now.
• Containers (i.e., JPanel) associated with a “layout” manager.
  – Determines how user interface elements are arranged within the container.
  – Different layout managers provide different types and styles of arrangement.
    • Some limit the number and location of component elements.
    • Differ in flexibility and sophistication
  – BorderLayout
    • Allows 5 components to be placed in one of 5 areas:
      – NORTH, SOUTH, EAST, WEST, CENTER
      – Center area is given any extra space.
      – Areas not filled are collapsed.

main_panel.setLayout(new BorderLayout());
Hello, World

```java
JLabel hello_world_label = new JLabel("Hello, World!");
hello_world_label.setHorizontalAlignment(SwingConstants.CENTER);
hello_world_label.setForeground(Color.BLUE);
hello_world_label.setBackground(Color.YELLOW);
```

• **JLabel**
  – Simple text component
  – Property setters for alignment, color, etc.
    • Colors in Java represented by class `java.awt.Color`
      – Predefined colors available as class constants.
  – **SwingConstants**
    • Defines a number of constants and enumerations used as symbolic names for several different methods.
Hello, World

main_panel.add(hello_world_label, BorderLayout.CENTER);

- Every user interface component is “contained” by some parent.
  - Here we add the label to the main_panel.
  - Second argument to add method depends on layout manager of container.
- Different layout managers need/support different kinds of arguments in order to specify where exactly newly added component will go.
Hello, World

main_frame.setVisible(true);

• Top-level window must be made visible.
  – Until now, Java was waiting until we had set everything up.
Top Level Windows

- **JDialog**
  - Dialog box top-level windows.
  - Several types pre-defined for ease of use.
    - File choosers, informational, input prompt, etc.

- **JFrame**
  - Normal top-level windows for UI

- **JApplet**
  - We’ll be ignoring this for now.
JOptionPane Examples

• JOptionPane provide quick and easy dialog boxes.
  – showConfirmDialog
    • Allows user to confirm a choice with a Yes/No/Cancel response.
  – showInputDialog
    • Prompts for text input.
  – showMessageDialog
    • Shows message, waits for user to acknowledge.

• All of these are “modal”.
  – Flow of execution is halted until dialog is dealt with.
  – Method returns response directly.

• lec18.v2
JFrame

• Non-dialog top-level window.
  – May or may not have a window bar
    • Location of which is operating system and look-and-feel specific
  – Operation defined by what is placed in it.
  – General pattern
    • Main method of program sets up UI in a JFrame
    • JFrame made visible.
    • Rest of program now responds asynchronously to user interface events.
    • Hello, world revisited.
Containment Hierarchy

• JPanel used as content pane for top-level window is root of “containment hierarchy”.
  – All user interface elements must be placed in a container (i.e., JPanel).
  – Containers can be nested within containers
    • Groups UI elements together and allows sub-layout.
Layout Managers

• BorderLayout
  – 5 areas: NORTH, EAST, WEST, SOUTH, CENTER
  – N/E/W/S also known as:
    • PAGE_START, LINE_START, LINE_END, PAGE_END

• BoxLayout
  – Stacked either horizontally or vertically

• GridLayout
  – Equal sized, regular grid.
Swing Components

• Text
  – JLabel, JTextField, JTextArea

• Buttons
  – JButton, JCheckBox, JRadioButton, JToggleButton

• Sliders
  – JSlider

• Lots of others
UI Events

• UI elements respond to interaction by generating “events”.
  – Listeners are registered to receive notification when a particular event occurs.
  – Different listener interfaces defined for different kinds of events.
    • Listening method receives event information as a parameter.

• UI Elements is like Observable
• Listener is like Observer
Button Example

• lec18.v3
• `addActionListener(ActionListener l)`
  – Method common to all types of buttons.
  – Defined in `AbstractButton`
    • Parent of all button types.
  – `ActionListener` interface:
    • `void actionPerformed(ActionEvent e)`
    • `ActionEvent` encapsulates all event information
  – `ActionEvent`
    • Parent class is `AWTEvent`
      – Common information for all event types.
      – `getSource()`
        » Returns reference to object that generated the event.
    • Other information provided specific to subclass.