Java GUI

COMP 401 Fall 2018

Lecture 17

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

- lec17.ex1

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.

Hello, World

```
main_panel.setLayout(new BorderLayout());
```

- 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.
      - Unfilled areas are collapsed.

Hello, World

```
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 understood by various 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.pack();
main_frame.setVisible(true);
```

- Top-level window must be made visible.
  - Until now, Java was waiting until we had set everything up.
  - Call to pack() resolves layout geometry.

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.
- lec17.ex2

JFrame

- Non-modal 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

- Recall that a JPanel object used as content pane for top-level window
  - Root of “containment hierarchy”.
  - All user interface elements must be placed in a container (e.g., JPanel).
  - Containers can be nested within containers
    - Groups UI elements together
    - Enables hierarchical 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.
- Should recognize this as observer/observable
  - UI elements are the observable
  - Listeners are the observers

Button Example

- lec17.ex3
- addActionListener(ActionListener l)
  - Method common to all types of buttons.
  - Defined in AbstractButton
    - Parent of all button types.
  - Serves as the observable's registration method.
- 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 depending on the subclass.

Java UI's so far

- Create a top level window.
- Create a JPanel to be content pane.
- Fill JPanel with other components
  - Buttons, sliders, etc.
  - Containers with components arranged in them.
    - And so forth.
- Connect UI events with "listeners"
  - Listeners take action as response.
  - Action may change/update UI
- Make top level window visible.

Swing Component Class Hierarchy

- java.awt.Component
  - java.awt.Container
    - javax.swing.JComponent

- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

- JComboBox
- JLabel
- JList
- JProgressBar
- JSeparator
- JSlider
- JToolTip
- JTree

- AbstractButton
  - JButton
  - JToggleButton
    - JCheckBox
    - JRadioButton
  - JMenuItem
    - JMenu

Listener Types

- Supported at awt.Component
  - ComponentListener
  - FocusListener
  - KeyListener
  - MouseListener
  - MouseMotionListener
  - MouseWheelListener
- Supported at awt.Container
  - ContainerListener
- Supported at AbstractButton
  - ActionListener
  - ChangeListener
  - ItemListener
- Supported by individual components
  - ActionListener : JComboBox
  - ChangeListener : JSlider, JProgressBar

Keyboard Concepts

- Keyboard events go to component that currently has “focus”.
  - setFocusable(boolean status)
  - isFocusable()
  - requestFocusInWindow()
- Java provides a framework for managing focus in more sophisticated ways if necessary.
  - http://docs.oracle.com/javase/tutorial/uiswing/misc/focus.html#api
- KeyListener interface
  - keyTyped(KeyEvent e)
  - keyPressed(KeyEvent e)
  - keyReleased(KeyEvent e)
- Typing of character vs. pressing of key
  - Basic sequence of events generated: key press, key typed, key released
- Important KeyEvent methods
  - char getKeyChar()
    This is only valid for key typed events.
  - int getKeyCode()
  - isAltDown(), isControlDown(), isShiftDown(), isMetaDown()

Mouse Concepts

- MouseListener
  - mouseClicked(MouseEvent e)
  - mouseEntered(MouseEvent e)
  - mouseExited(MouseEvent e)
  - mousePressed(MouseEvent e)
  - mouseReleased(MouseEvent e)
- MouseMotionListener
  - mouseDragged(MouseEvent e)
  - mouseMoved(MouseEvent e)
- MouseEvent
  - Position info
    - Relative to component: getX(), getY(), getPoint()
    - Absolute position: getXOnScreen(), getYOnScreen()
  - Click info: getClickCount(), getButton()
  - Modifiers: isAltDown(), isShiftDown(), etc...

Composing Widgets

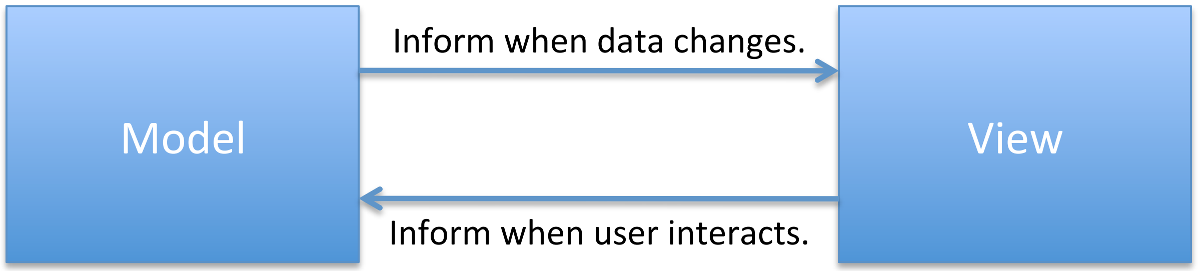
- A set of related UI elements that act as a unit within your UI.
- Basic idea:
  - Subclass JPanel
  - In constructor, create and arrange UI components.
  - Provide methods for attaching listeners, requesting current state of UI, etc.
    - Could be straight delegation or could be mediated by more sophisticated logic.

Color Chooser Widget

- lec17.ex4.v1
  - Basic widget construction and set up
- lec17.ex4.v2
  - Wiring up internal behavior
- lec17.ex4.v3
  - Providing external behavior
- lec17.ex4.v4
  - Demonstrating keyboard concepts

Model – View Pattern

- Keep object representing the data separate from widget providing the UI
  - Model: Object representing logical entity within your application.
  - View: Object (widget) representing UI to model.


Model
Inform when data changes.
View
Inform when user interacts.