Chapter 6: Graphical User Interfaces

CS 121

Department of Computer Science
College of Engineering
Boise State University

November 2, 2015

Chapter 6 Topics

- Anatomy of a Graphical User Interface (GUI) Go to part 0

Anatomy of a GUI (1)

- An application with a Graphical User Interface (GUI) allows an user to interact with the application in multiple ways unlike a command line application.
- A GUI consists of three types of objects: components, events and listeners
  - Component: An object that defines a screen element for displaying information or interacting with the user. For example: a button, a label, a text field etc.
    - Container: A container is a special type of component that is used to hold and organize other components. For example: a frame, a window, a panel etc.

Anatomy of a GUI (2)

- Event: An object that represents an occurrence we are interested in.
  - Button clicked, mouse pressed, mouse moved, keyboard key pressed, timer expired etc.
  - Most GUI components generate an event to indicate an user action related to that component.
  - Applications that respond to events from a GUI are examples of event-driven applications.
- Listener: An object that *waits* for an event to occur and responds when it does.
- In a GUI program, we need to establish the relationships between between the listener, the event it listens for, and the component that generates the event.

Anatomy of a GUI (3)

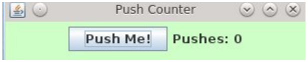
- To create a Java program that contains a GUI, we must:
  - instantiate and set up the necessary components,
  - implement listener classes that define what happens when particular events occur, and
  - establish the relationship between the listeners and the components that generate the events of interest.
- Components, events and related classes are primarily defined in two Java packages:
  - `java.awt`: The original *Abstract Windowing Toolkit* GUI package that contains many important classes.
  - `javax.swing`: The *Swing* package was added later and is more versatile. It builds upon the AWT package.

Containers: Frames and Panels

- Containers are classified as either
  - heavyweight – managed by the underlying operating system. For example: a frame.
  - lightweight – managed by the Java program. For example: a panel.
- A standalone GUI application creates a frame as its main window. A frame contains a titlebar, with buttons to resize and close the window. The frame object in Swing is called a `JFrame`.
- Examples: FrameExample1.java, FrameExample2.java
- The `JFrame` contains four panes: *Root Pane*, *Layered Pane*, *Content Pane*, and the *Glass Pane*. We will only be using the Content Pane.
- Typically, we will have the frame contain a panel that contains all the other components of our application. This allows our program to be more independent of the underlying operating system.

Example 1: A Complete Simple GUI

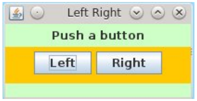
- Let's look at a simple example that contains all of the basic GUI elements
  - the GUI presents the user with a single push button
  - each time the button is pushed, a counter is updated and displayed



- The example uses the following
  - Components: `JFrame`, `JPanel`, `JButton`, `JLabel`.
  - Events: `ActionEvent` (generated when a button is pushed or clicked)
  - Listener: We write our own class that `implements` the `ActionListener` interface to react to the events.
- Example: PushCounter.java, PushCounterPanel.java
- Also see example: PushCounterPanel2.java

Example 2: Listening to Multiple Components

- We can use one listener to listen to two different components.
- For example: we have one label and two buttons
  - when the Left button is pushed, the label displays "Left"
  - when the Right button is pushed, the label displays "Right"



- Example: LeftRight.java, LeftRightPanel.java
- Now the `actionPerformed` method gets called for either button press. We use the `getSource()` method in the `ActionEvent` object to determine which button was pressed.

More Components

- `JTextField`: A text field that allows an user to enter input typed from the keyboard on a single line.
- `JTextArea`: A text area is a multi-line version of a text field.
- `JScrollPane`: A scroll pane provides a scrollable view of a component. For example, for a text area with more text than fits in the display.
- `JCheckbox`: A button that can be toggled on or off.
- `JRadioButton`: Used with a group of radio buttons to provide a set of mutually exclusive options.
- `JSlider`: Allows the user to specify a numeric value within a bounded range.
- `JComboBox`: Allows the user to select one of several options from a "drop down" menu.
- `Timer`: Allows us to animate or automate things. Has no visual representation.

JTextField

- A text field generates an action event (`ActionEvent`) object when the `Enter` or `Return` key is pressed in the text field.
- Note that the push button and the text field generate the same kind of event — an action event.
- Example: Fahrenheit.java, FahrenheitPanel.java

JTextArea and JScrollPane

- JTextArea is a component that allows text to be displayed. The text can be set to be editable or not editable.
- A `JScrollPane` can manage a `JTextArea` to display scroll bars as needed or always.
- See sample code below:

```java
private JTextArea display = new JTextArea(10,20);
display.setEditable(false);
JScrollPane scroller = new JScrollPane(display,
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
```

- Example: TextAreaTest.java

Layout Managers (1)

- A layout manager determines how the components in the container are arranged visually. A layout manager determines the size and position of each component
- The layout manager is consulted when needed, such as when the container is resized or when a component is added.
- Every container has a default layout manager, but we can replace it if desired.

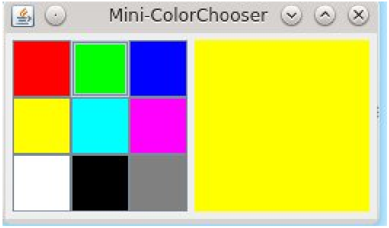
Layout Managers (2)

- Some of the layout managers in the Java API:

Layout Manager	Description
`FlowLayout`	Puts components from left to right, starting new rows as needed
`GridLayout`	Puts components into a grid of rows and columns
`BoxLayout`	Puts components into a single row or column
`BorderLayout`	Puts components into five areas (North, South, East, West, Center)

- Example: LayoutDemo.java, IntroPanel.java FlowPanel.java, GridPanel.java BoxPanel.java, BorderPanel.java

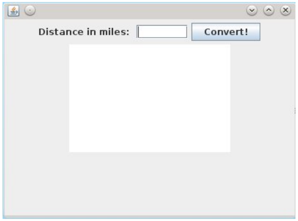
A Bigger Example



- Uses an array of buttons laid out in a grid.
- The frame is divided into two panels: one for the grid of buttons, the other for a panel that shows the chosen color.
- Clicking a button sets the color of the panel to the right of the buttons panel.
- Example: MiniColorChooserV1.java

Design example 1: Converter App (1)

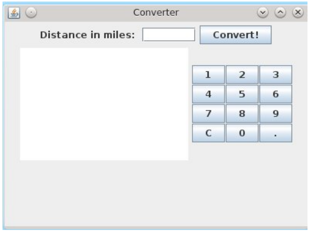
- We will develop a series of GUI to illustrate several design principles.
- The GUI is a simple app that converts miles to kilometers and displays the result.



- Example: converter1/Converter.java, converter1/MetricConverter.java
- Use the Model View Controller(MVC) design.
  - *Model*: MetricConverter.java
  - *View*: Setup of the GUI in the constructor for Converter.java
  - *Controller*: The listener code in Converter.java
- Identify improvements to the user interface.

Design example 1: Converter App (2)

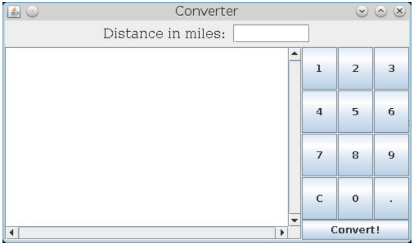
- We will add an `ActionListener` to the text field so that the user can use the app with only the keyboard
- We will add an input keypad so that the user can use the app with only a mouse.



- Example: converter2/Converter.java, converter2/MetricConverter.java
- Identify further improvements to the user interface.

Design example 1: Converter App (3)

- Add better layout to the main panel.
- Add a scroll pane to the display area.
- Organize the code more. Add a private class for the Controller.
- Catch the `NumberFormatException` on the text field and report error to user.



- Example: converter3/Converter.java, converter3/MetricConverter.java
- What else would you like to improve?

More Components: Check Boxes

- A check box generates an item event when it changes state from selected (checked) to deselected (unchecked) and vice versa. The `JCheckBox` class is used to define check boxes.
- They produce `ItemEvent` events that use an `ItemListener` interface, which has one method :

  ```
  public interface ItemListener {
      public void itemStateChanged(ItemEvent event);
  }
  ```

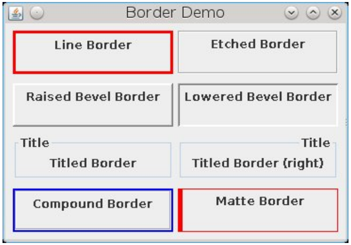
- Example: StyleOptions.java, StyleOptionsPanel.java

More Components: Radio Buttons

- A radio button is used with other radio buttons to provide a set of mutually exclusive options.
- Radio buttons have meaning only when used with one or more other radio buttons At any point in time, only one button of the group is selected (on).
- Radio buttons produce an action event when selected Radio buttons are defined by the `JRadioButton` class. The `ButtonGroup` class is used to define a set of related radio buttons.
- Example: QuoteOptions.java, QuoteOptionsPanel.java

Borders

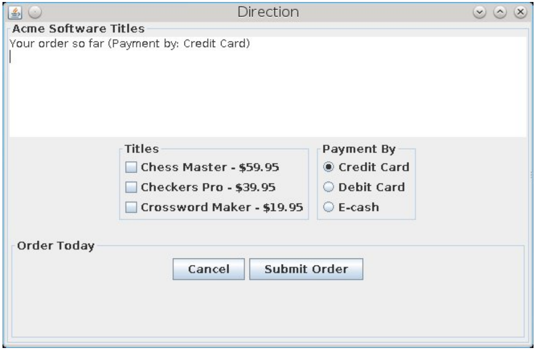
- Java provides the ability to put a border around any Swing component.
- Border provide visual cues as to how GUI components are organized.
- The `BorderFactory` class is useful for creating borders for components.



- Example: BorderDemo.java

Design Example 2: Order Application

- This example illustrates layouts, borders, radio buttons and check boxes.



Your order so far (Payment by: Credit Card)

☐ Chess Master - $59.95
☐ Checkers Pro - $39.95
☐ Crossword Maker - $19.95

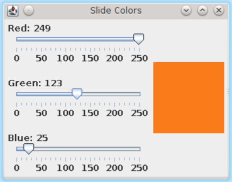
● Credit Card
○ Debit Card
○ E-cash

- **In-class Exercise**: Sketch the various containers, components and borders needed to produce this layout.
- Example: In the package orderapplication, see OrderApplication.java, OrderApplicationPanel.java.

More Components: Sliders

- A slider can be presented either vertically or horizontally. Optional features include:
  - tick marks on the slider,
  - labels indicating the range of values.
- A slider produces a change event, indicating that the position of the slider and the value it represents has changed.
- A slider is defined by the `JSlider` class. It produces `ChangeEvent` events that require a `ChangeListener` interface.

```
public interface ChangeListener {
    public void stateChanged(ChangeEvent event);
}
```



- Example: SlideColor.java, SlideColorPanel.java

More Components: Combo Boxes

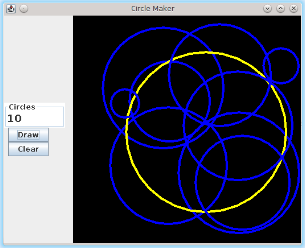
- A combo box allows a user to select one of several options from a “drop down” menu.
- When the user presses a combo box using a mouse, a list of options is displayed from which the user can choose.
- A combo box is defined by the `JComboBox` class. Combo boxes generate an action event whenever the user makes a selection from it.
- Example: JukeBox.java, JukeBoxPanel.java

Timer

- Timers are defined by the `Timer` class and are provided to help manage an activity over time.
- A timer object generates an action event at regular intervals.
- Example: Rebound.java, ReboundPanel.java

Design Example 3: CircleMaker

- Design a GUI that allows the user to make the specified number of circles and color the biggest circle with a different color.



- Model: Circle.java
- View: CircleMakerPanel.java
- Controller: CircleMaker.java

More Events:Mouse and Mouse Motion

- Mouse actions generate `Mouse Event` objects.
- Two types of interfaces to deal with mouse events:
  - mouse events – occur when the user interacts with another component via the mouse: *pressed, clicked, released, entered, exited*. To use, implement the `MouseListener` interface class
  - mouse motion events – occur while the mouse is in motion: *moved*, *dragged*. To use, implement the `MouseMotionListener` interface class.

Mouse Examples

- Example:Dots.java, DotsPanel.java
  - Clicking the mouse causes a dot to appear in that location and the coordinates to be displayed. Overall count of all the dots is also displayed.
  - The event object passed to the listener is used to get the coordinates of the event.
  - An `ArrayList` is used to keep track of the points.
- Example:RubberLines.java, RubberLinesPanel.java
  - As the mouse is dragged, the line is redrawn. This creates a rubberbanding effect, as if the line is being pulled into shape.
- **In-class Exercise**: Write a mouse odometer that displays (in pixels) how far the mouse has traveled!

More Events: Keys

- A key event (`KeyEvent` object) is generated when the user presses a keyboard key This allows a program to respond immediately to the user while they are typing.
- The `KeyListener` interface defines three methods used to respond to keyboard activity:

```
public interface KeyListener {
    public void keyPressed(KeyEvent event);
    public void keyReleased(KeyEvent event);
    public void keyTyped(KeyEvent event);
}
```

- Example: Direction.java, DirectionPanel.java
- **In-class Exercise**: Modify the example above so that the key image wraps around in either of the four directions.

Dialog Boxes

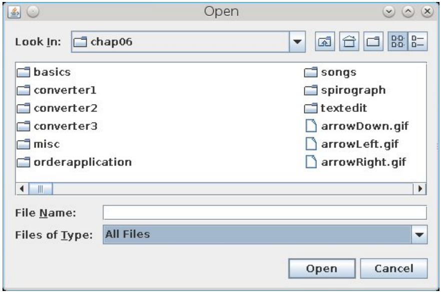
- A dialog box is a window that appears on top of any currently active window. A dialog box usually has a specific, solitary purpose, and the user interaction with it is brief. It may be used to:
  - convey information (`JOptionPane`)
  - confirm an action (`JOptionPane`)
  - allow the user to enter data (`JOptionPane`)
  - pick a color (`JColorChooser`)
  - choose a file (`JFileChooser`)

JOptionPane

- `JOptionPane` dialog boxes fall into three categories:
  - message dialog boxes – used to display an output string.
  - input dialog boxes – presents a prompt and a single input txt file into which the user can enter one string of data.
  - confirm dialog box – presents the user with a simple yes-or-no question.
- These three types of dialog boxes are created using static methods in the JOptionPane class
- Example: EvenOdd.java

JFileChooser

- A file chooser is a specialized dialog box used to select a file from a disk or other storage medium.
- The dialog automatically presents a standardized file selection window.
- Filters can be applied to the file chooser programmatically.
- The `JFileChooser` class creates this type of dialog box.



- Example: FileChooser.java

More Components

- A color chooser class: `JColorChooser`. See example SlideColor.java, SlideColorPanel.java
- A clickable list class: `JList`. See example basics/ClickableListDemo.java
- A split window example using `JSplitPane`: basics/SplitWindows.java
- A menu example using `JMenuBar`, `JMenu` and `JMenuItem`: basics/MenuDemo.java

Tooltips, Mnemonics

- A tool tip is a short line of text that appears over a component when the mouse cursor is rested momentarily on top of the component.
- Tool tips can be assigned by using the `setToolTipText` method of a component.

```
JButton button = new Button("Compute");
button.setToolTipText("Calculates the area under the
    curve");
```

- A mnemonic is a character that allows the user to push a button or make a menu choice using the keyboard in addition to the mouse.
- The user can hold down the *Alt* key and press the mnemonic character to activate (depress) the button. We set the mnemonic for a component using the `setMnemonic` method of the component.
- Example: LightBulb.java, LightBulbControl.java, LightBulbPanel.java.

Design Example 4: Choose Your Adventure!

- A detailed inclass case study of a GUI chosen by our instructor....

Summary

- Containers: JFrame, JPanel, JDialog, JWindow
- Components: JButton, JLabel, JTextField, JCheckBox, JRadioButton, ButtonGroup JTextArea, JSlider, JColorChooser, JFileChooser, JOptionPane, JTabbedPane, JScrollPane, JSplitPane, JList, JMenu, JMenuItem, JMenuBar
- Layout Managers: FlowLayout, BorderLayout, GridLayout, GridBagLayout, BoxLayout, CardLayout
- Events: ActionEvent, ItemEvent, WindowEvent, MouseEvent, KeyEvent
- Listeners: ActionListener, ItemListener, MouseListener, MouseMotionListener, KeyListener

Exercises

- Read Chapter 6.
- **Recommended Homework**:
  - Exercises: EX 6.3, 6.6, 6.9, 6.10.
  - Projects: PP 6.5, 6.16, 6.21, 6.22.