



- GUIs
- Buttons
- Text Fields
- More GUI stuff!
- Dialog Boxes
- Check Boxes
- Radio Buttons



- So far, we saw two fundamental elements for creating a GUI: components and containers
 - We've previously discussed components, which are objects that represent screen elements
 - E.g., labels, buttons, text fields, menus
 - Some components are containers that hold and organize other components
 - E.g., frames, panels, applets
- We are missing an important piece:
 - events
 - listeners



- An *event* is an **object** that represents some activity to which we may want to respond
 - E.g., the mouse is moved, a graphical button is clicked, a keyboard key is pressed
- Events often correspond to user actions, but not always (e.g., timer)
- The Java library contains several classes that represent typical events
- Components generate (or fire) events



- A *listener* is an **object** that "waits" for an event to occur and responds accordingly
- We can design listener objects to take whatever actions are appropriate



- When the event occurs
 - the component calls the appropriate method of the listener
 - passing an object that describes the event



- **To create a GUI-based Java program we must:**
 - Instantiate and set up the necessary components
 - Identify the events we care about
 - Implement listener classes for those events
 - Establish the relation between listeners and components that generate the corresponding events
- **We illustrate with two components**
 - Push button: JButton
 - Text field: JTextField



- **When pressed, it generates an `ActionEvent`**
- **Listeners must implement `ActionListener` and define method `actionPerformed(ActionEvent e)`**
 - The only method in the `ActionListener` interface is the `actionPerformed` method
- **Method `addActionListener` is used to specify listeners**
 - `addActionListener(ActionListener listener)`
- **`PushCounter` example**



- GUI components: button, label (counter), panel, main frame
- **PushCounterPanel**
 - See constructor
 - See inner class
 - Defined within the body of another class
 - Facilitates the communication between the listener and the GUI components
 - Used in situations where there is an intimate relationship between the two classes and the inner class is not needed in any other context
 - See call to **addActionListener**



- When "enter" is pressed, it generates an **ActionEvent**
- Method **addActionListener** is used to specify listeners
 - **addActionListener(ActionListener listener)**
- Listeners must implement **ActionListener**
- **Fahrenheit** example
(analogous to previous one)



- See [LeftRight.java](#) and [LeftRightPanel.java](#)
- Works, but...
 - One listener object can be used to listen to two different components
 - The source of the event can be determined by using the **getSource** method of the event passed to the listener
 - How do we change the code then?
 - What if we added a second handler?



- A dialog box is a window that appears on top of any currently active window and can be used to
 - convey information, confirm an action, read user data, ...
- Specific, solitary purpose, brief user interaction
- Jframes could be used, but they are non-modal!
- Class `JOptionPane` provides methods that simplify the creation of some types of d.b -- See [EvenOdd.java](#)
- How can we change the listener so that dialog boxes take the frame as the parent?



- Button that can be toggled on or off
- Represented by the JCheckBox class
- Generates an ItemEvent whenever it changes state (on/off)
- Listener: ItemListener, itemStateChanged
- [StyleOptions.java](#): Uses CehekBoxes to set the style of a label's text
- **Let's do the listener using class Font:**
 - family name (such as Times or Courier)
 - style (plain, bold, italic, or bold|italic)
 - Size



- **Grouped, represent mutually exclusive options**
 - When one is selected, the one that is "on" goes "off"
- Use ButtonGroup to define the group
- A radio button generates an ActionEvent
- [QuoteOptions.java](#): Uses RadioButtons to determine which text line to display
- Differences between check boxes and r.b.?



- Conditionals and loops enhance our ability to generate interesting graphics
- See [Boxes.java](#) and [BoxesPanel.java](#)
- Problem:
 - What if we wanted to color the largest box in red?