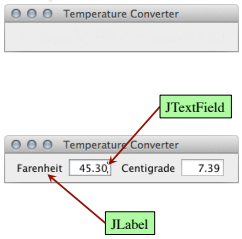
The Limitations of JFrame

- JFrame is just a Window
  - Can resize it
  - Can close it
  - Not much else
- To do more, you need **GUI components**
  - Items inside a JFrame
  - Ex: Buttons, Text Boxes
- Two main Java packages
  - java.awt: "old GUI"
  - javax.swing: "Swing GUI"



Swing Components

`JButton`: a pushbutton that can be clicked by mouse
`JCheckbox`: can be on (true) or off (false)
`JComboBox`: a popup menu of user choices
`JLabel`: a text label
`JList`: scrolling list of user-chooseable items
`JScrollbar`: a scroll bar
`JTextField`: allows editing of a single line of text
`JTextArea`: multiline region for displaying and editing text
`JPanel`: used for containing and grouping components
`JDialog`: window used for user input
`JFrame`: top-level window with frame and border
…

Buttons

List

Slider

Main Challenges in GUI Applications

Layout

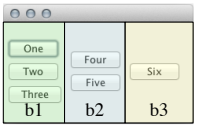
- Arranging items the screen
  - Java has many components
  - But where do they go?
- **Challenge**: Resizing
  - Want components to "behave nicely" as you resize
  - Change size of components
  - Change padding in between
- LayoutManagers do both

Input Handling

- Many types of input
  - button pushed
  - text typed
  - mouse clicked …
- Want app to react to input
  - Otherwise GUI looks pretty, but does nothing
- **Main focus of GUI code**

BoxLayout: The Best for Beginners

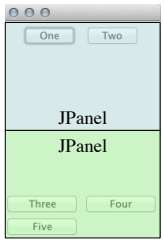
- BoxLayout
  - Arranges components in line
  - No wrap (like FlowLayout)
  - Either horizontal/vertical
- Box: JPanel w/ BoxLayout
  - Box b1= **new** Box(BoxLayout.Y_AXIS);
  - Makes layout quick
- **Demo**: BoxGrouping.java



- Nested boxes
  - Three vertical boxes
  - Inside horizontal box

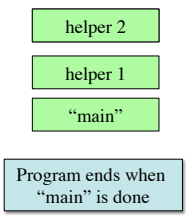
Nesting Layouts

- Want more interesting layouts
  - **Idea**: nest layouts in each other
  - Can get fine padding control
- Useful class: JPanel
  - Invisible component
  - Container for other components
  - Can take a LayoutManager
- **Demo**: PanelGrouping.java



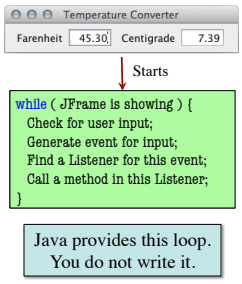
Traditional Programming

- Have a "main" method
  - Call in Interactions pane
  - Call in JUnit test
  - …somewhere else?
- Other methods are helper methods to "main" one
- Big reason for DrJava
  - Usually only one "main"
  - Interactions pane allows all methods to be "main"

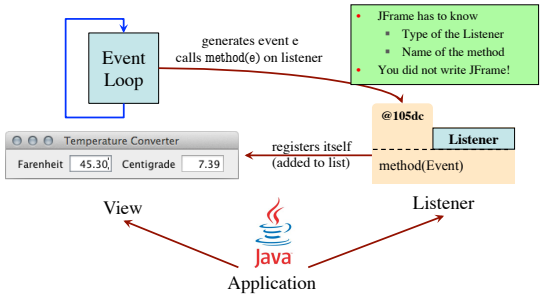


Listeners

- A Listener is a class with methods to respond to input
  - ImageProcessor in A6
  - Each method is a GUI button
  - Support other types of input
- Program registers Listeners with an event type
  - Event loop finds a Listener for the current event type
  - Calls a Listener method
  - Event is passed as argument

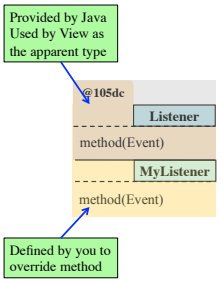


Event-Driven Programming



Solution: Apparent Types

- Java provides a Listener type
  - Has the method already in it
  - Subclass this as your own class
  - Override method for your usage
- View uses the Listener type
  - Allows it to call the method
  - Uses your version of method (bottom-up rule)
- Designed to be overridden…



Abstract Classes: Made to be Overridden

- Abstract method
  - Has the method header
  - But does not have body!
  - Example: Piece.java
- Why do this?
  - Will use Piece for the apparent type (variable)
  - But Piece will never be the real type of anything
- Artifact of static typing

```
public class Piece {
  ...
  // Abstract
  public abstract void
                act(JManBoard board);
}

public class JMan {
  ...
  // IMPLEMENTATION
  public void act(JManBoard board) {
    ...
  }
}
```

Listeners are actually Interfaces

- Like an abstract class
  - But **all methods** abstract!
  - And cannot have fields
- What is the difference?
  - Don't **extend** an interface
  - You **implement** one
- What the heck????
  - Major topic in CS 2110
  - Not needed for JMan
  - We did this for you

```
public interface A {
  public void doIt();  // Abstract
}

public class B implements A {
  public void doIt() {
    ...
  }
}
```

Listeners and Events in Java

In packages:
- javax.swing.event
- java.awt.event

Events	Listeners
• ActionEvent ▪ User clicks a button ▪ User hits return in text field	• ActionListener ▪ actionPerformed(ActionEvent)
• MouseEvent ▪ User clicks the mouse ▪ User moves the mouse	• MouseListener ▪ mouseClicked(MouseEvent) ▪ mouseEntered(MouseEvent)
• KeyEvent ▪ User presses a key ▪ User releases a key	• MouseMotionListener ▪ mouseDragged(MouseEvent) • KeyListener ▪ keyPressed(KeyEvent)