CSE11 – Lecture 13
Fall 2013
Java AWT Intro

Graphical Programs in Java AWT and Swing

- Three Major Topics for Today's Lecture

1) Graphical Components
   - Buttons, Menus (ComboBox), Sliders, Labels, Input Fields, Multi-line text

2) Layout Managers
   - What they (at least 8 different ones!) are for and why
   - How to work with a couple of them

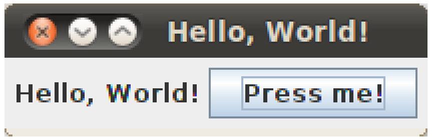
3) Events
   - Click a mouse, type on the keyboard, move slider … how do let your objects know that these have occurred?

Why?

- We've been using objectdraw – it's a good learning tool.
  - It simplifies our life to be able to do some interesting things
  - There is a rich world of Java tools to use, it's time to learn a little bit about them
- Java is pretty good for these more complicated (so-called) “rich” web interfaces

Java Swing

- `import javax.swing.*;`
- It's a general GUI (Graphical User Interface) builder in Java.
- It separates the definition of the interface with the actual presentation
  - You define various objects that you want to place within a Frame
  - Swing decides exactly how to present them
  - Your window manager (or other programs) can decide how to color and decorate the Frame.



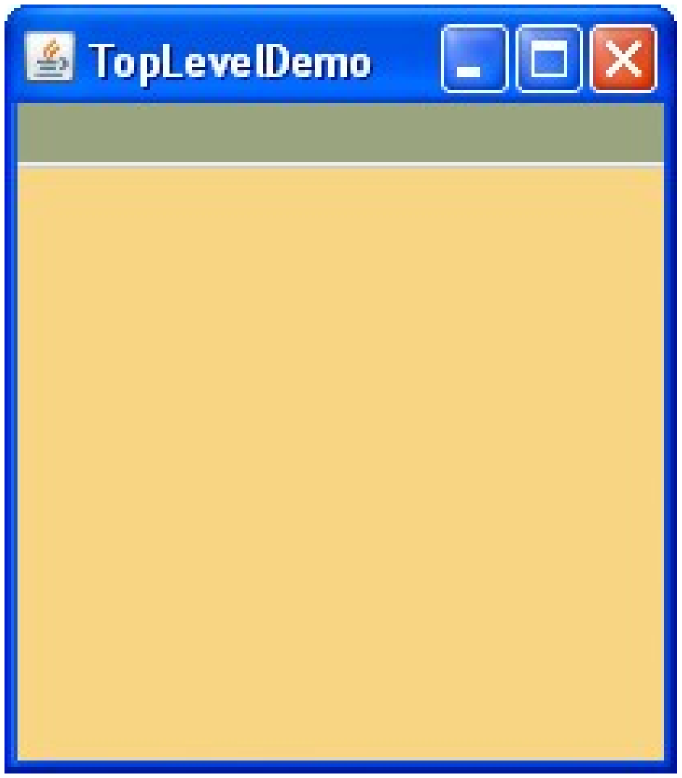
Hello World using Swing

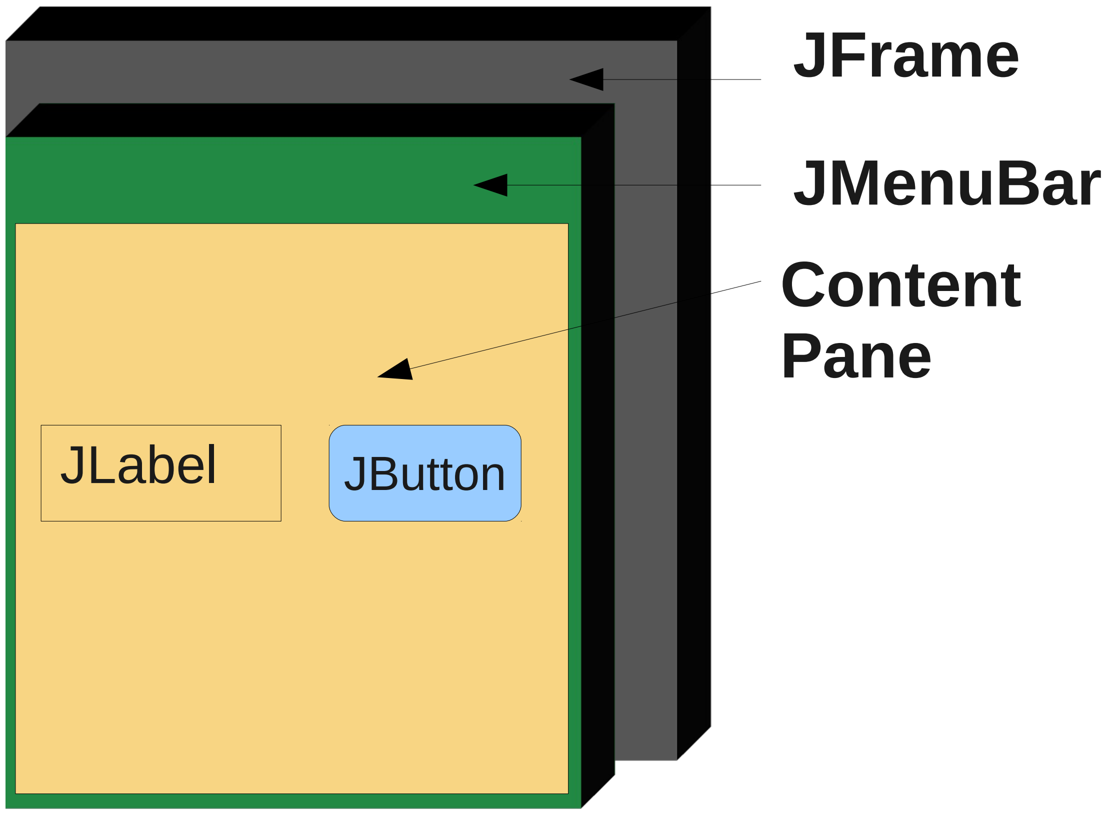
```
// Compile: javac HelloWorldSwing.java
// Run: java HelloWorldSwing
import java.awt.*;
import javax.swing.*;

public class HelloWorldSwing implements Runnable {
    public void run() {
        // Create the window, Set behavior when closed
        JFrame f = new JFrame("Hello, World!");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Get a reference to the Frame's content pane
        Container pane = f.getContentPane();
        // Add a layout manager so that the button is not
        // placed on top of the label
        pane.setLayout(new FlowLayout());
        // Add a label and a button
        pane.add(new JLabel("Hello, World!"));
        pane.add(new JButton("Press me!"));
        // Arrange the components inside the window
        f.pack();
        // By default, the window is not visible. Make it visible.
        f.setVisible(true);
    }
    public static void main(String[] args) {
        HelloWorldSwing hws = new HelloWorldSwing();
        // Schedules the application to be run time in the event queue.
        SwingUtilities.invokeLater(hws);
    }
}
```

Swing is Component-based

- http://docs.oracle.com/javase/tutorial/uiswing/components/frame.html
- JFrame : Main Window. For our Purposes it holds (or contains) all other Swing Objects in its content pane
- JButton : A button that can be pressed with the mouse
- JLabel: Text that is displayed

JFrame is Top Level Container



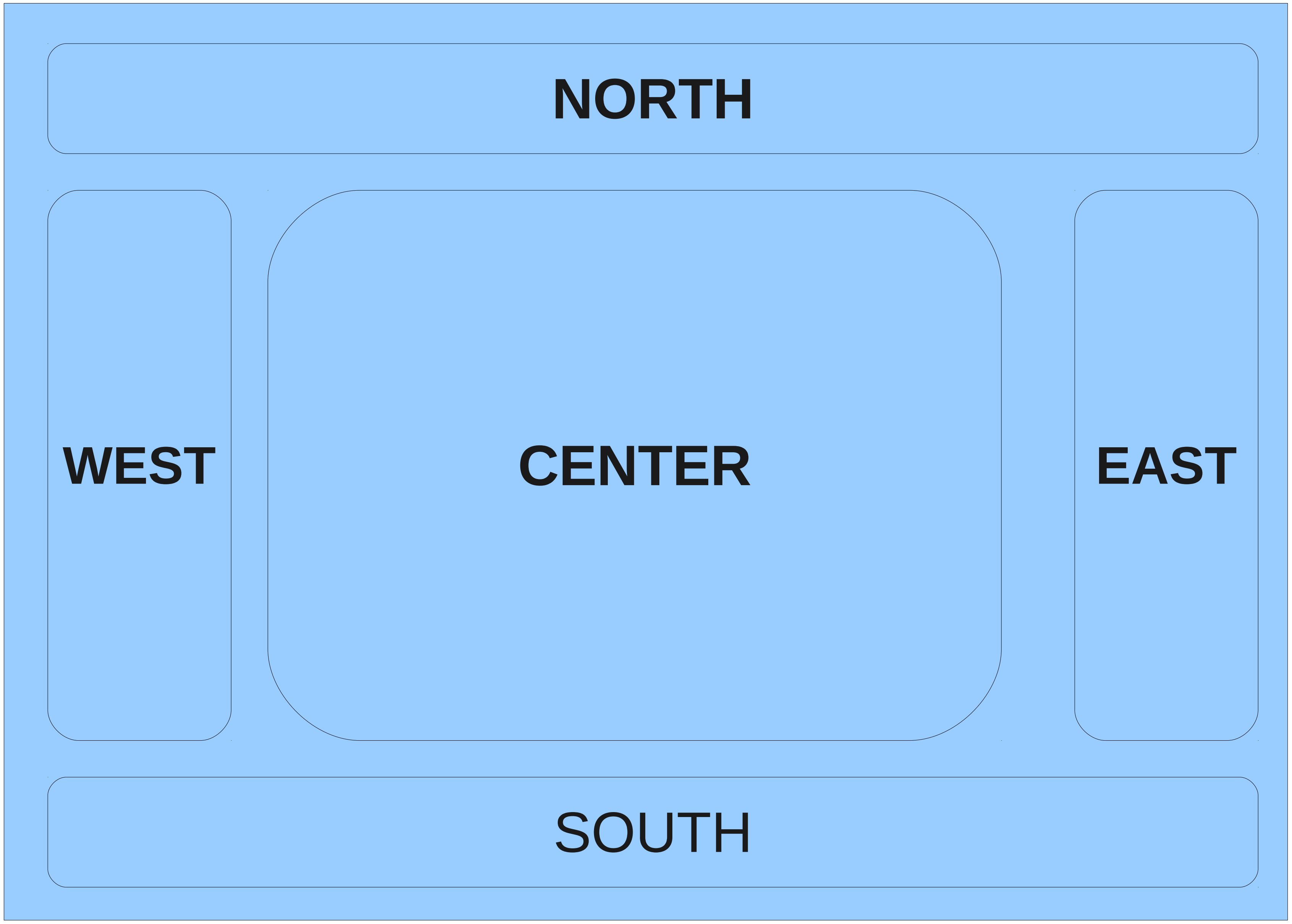


- Containers have a content pane where GUI components are displayed
- Containers can form a Hierarchy of Containers

Layout Managers

- In Swing, GUI components are placed on the content pane under the control of a layout manager
  - Note we did not specify X,Y coordinates for where to place the JButton or Jlabel
- There are 8 different layout managers in Swing
  - BorderLayout, FlowLayout, GridLayout, ...
- Let's look at BorderLayout.

Border Layout



Objectdraw Relationship to JFrame (and JApplet)

- WindowController is a `Container` object
  - It inherits from `JApplet`
  - All Containers have content panes
  - getContentPane() returns the object the represents the Content pane of a container
- WindowController futher sets the content pane's LayoutManager to be `BorderLayout`
- Finally, the canvas is placed into the center area of the BorderLayout

Look at First Book Example

- http://eventfuljava.cs.williams.edu/sampleProgs/ch11/textb
- `JTextField` is an object where the user can type in input
- Constructs a `JTextField` instance and places it in the `BorderLayout.NORTH` area of the content pane
- Let's make a modifications to this example and see what happens
  - Change the Layout to FlowLayout

Events

- Java is a pretty good language/environment for GUIs
- There are certain (Swing) objects the generate Events (e.g, `ActionEvents, ChangeEvents, KeyEvents MouseEvents, ...)` when something happens
  - Button pushed, text entered, menu item selected, slider moved
- Other objects can register that they are interested in being notified when a particular Eventt Occurs
  - .e.g. Tell me when the “OK!” button was pushed
  - Tell me when a slider is moved, etc.

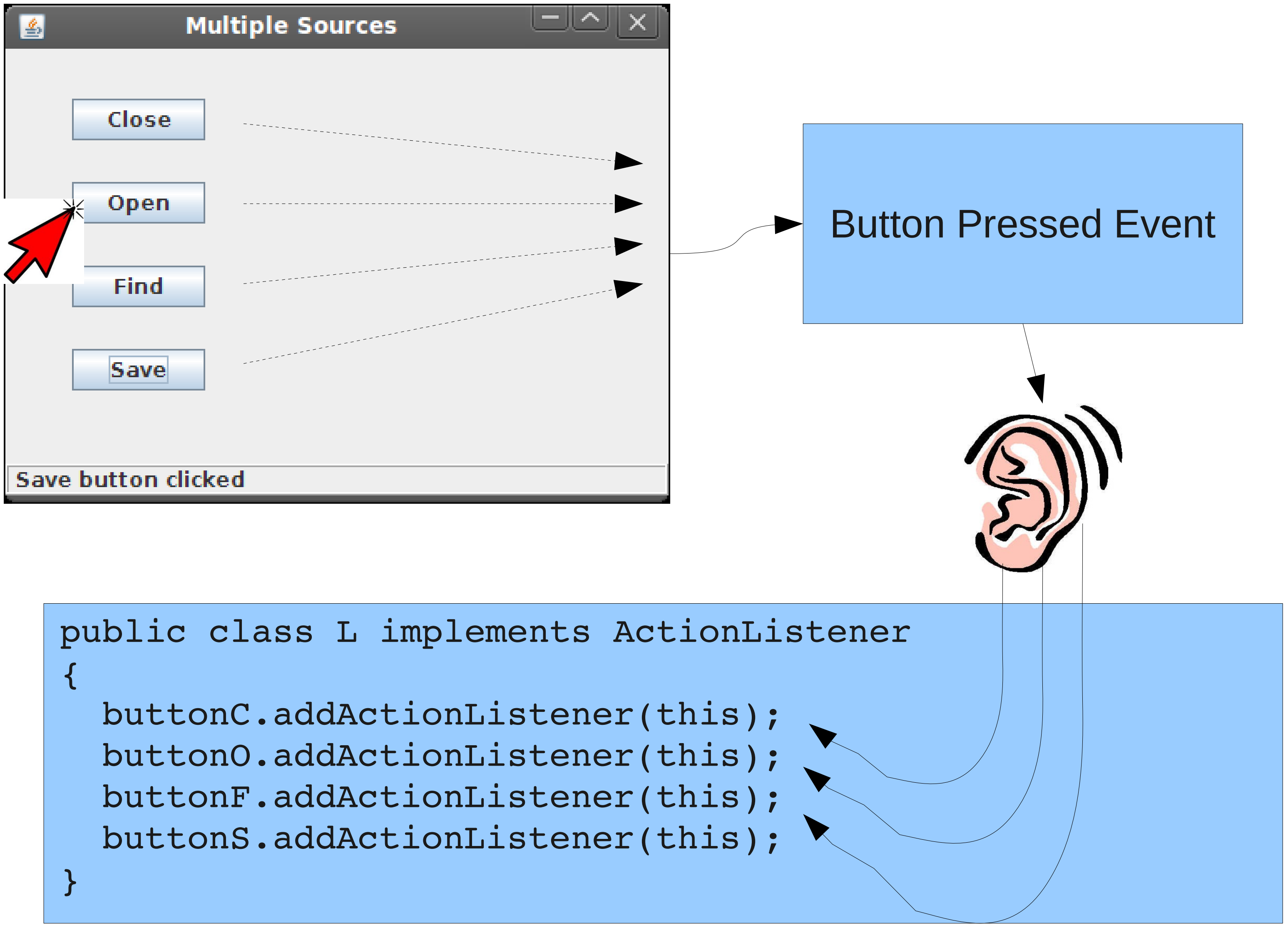
ActionListener

- A class that wants to react to action events implements the **`ActionListener` Interface**
  - Must define a `public void actionPerformed( .. )` method
- The class then registers with the particular object (e.g. a button or a slider) that it will listen for (and process) it's events
  - `object.addActionListener( ActionListener C)`

Modify HelloWorldSwing.java --> HelloWorldSwingCount.java

- Add another JLabel that prints out how many times we have pressed the Button
- Add that it implements the `ActionListener` interface
- Define `actionPerformed()` method
  - Increments count of button pushes
  - Updates the text of the new Jlabel
- NOTE: onMouseClick(), onMouseDrag(),... are defined by objectdraw and work only in objectdraw's canvas. You can't use them to handle JButton (or other events)

Multiple Buttons



```
public class L implements ActionListener
{
  buttonC.addActionListener(this);
  buttonO.addActionListener(this);
  buttonF.addActionListener(this);
  buttonS.addActionListener(this);
}
```

actionPerformed (ActionEvent evt)

```
public actionPerformed (ActionEvent evt)
{
    if (evt.getSource() == buttonC )
    {
        // close button pressed
    }
    else if (evt.getSource() == buttonO)
    {
        // Open button pressed
    }
    else if (evt.getSource == buttonF )
    {
        // Fine button pressed
    }
    else if (evt.getSource == buttonS )
    {
        // Save button pressed
    }
}
```

Panels

- Suppose we want to use the BorderLayout, but wanted two buttons to be across the bottom (BorderLayout.SOUTH) position?
- If we just add 3 buttons, they will be one right on top of the other.
- Swing Solution:
  - Create a panel (JPanel), define the Layout of the Panel to be FlowLayout
  - Add the panel to the BorderLayout.SOUTH
- A panel is a container, too.
- Example: TextControllerPanel.java
-

JcomboBox'es

- JComboBox
  - This is a menu/choice selection feature of Java
  - Let's change the TextControllerPanel to use a combo box instead
  - See TextControllerCombo

  Note: In versions of Java after the book was written, JComboBox is a “generic” type and needs to be told what kind of objects the combo box will hold. `JcomboBox<String>`

Other GUI Components

- `JTextArea.` A multiline Text input/output object.
  - Can be made scrollable in the horizontal and vertical dimensions
  - `new JTextArea (initialContents,  #Rows, #Cols)`
- `JSlider` – Create horizontal or vertical sliders with min, max, initial integer values
  - implement `ChangeListener` Interface
    - `public void stateChanged( ChangeEvent evt)`
    - `slider.getValue()` to read the new value
  - use `slider.addChangeListener(this)`
    - instead of `addActionListener`

Mouse and Keyboard events

- to Listen for Keyboard events, implement `KeyListener` interface
  - keyPressed(KeyEvent evt)
  - keyRelease(KeyEvent evt)
  - keyTyped(KeyEvent evt)
- To listen for mouse events implement `MouseListener` and/or `MouseMotionListener` interfaces