Object Oriented Programming and Design in Java

Session 5
Instructor: Bert Huang

Announcements

- Homework 1 due Feb. 17th 11 AM
  - 9 days

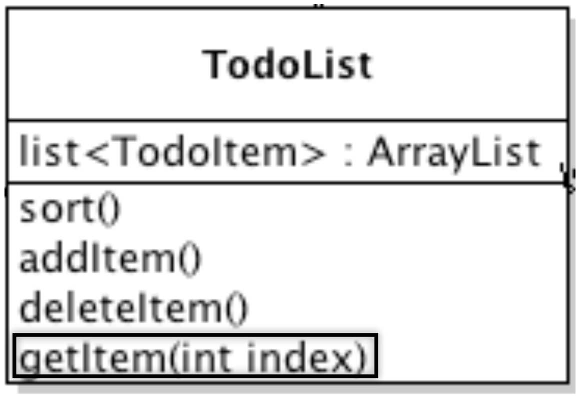
Review

- Review example from end of last class
- Designing classes
  - encapsulation
  - accessors/mutators
  - programming by contract: preconditions, postconditions, invariants

Today’s Plan

- Introduction to Java graphics
  - Swing classes: JFrame, JComponent, JButton, JTextField, JPanel
  - ActionListener interface
  - Graphics: Graphics2D

ToDoList.getItem()



- getItem(int index)
- ~~@precondition 0 ≤ index < list.size()~~
- @postcondition list is sorted
- @throws `IndexOutOfBoundsException`
- (This design is flawed.)

Three Notions of Interfaces

- Set of public methods
- Abstract Java class, containing a set of public methods
- User interface: how users provide input and how programs provide feedback

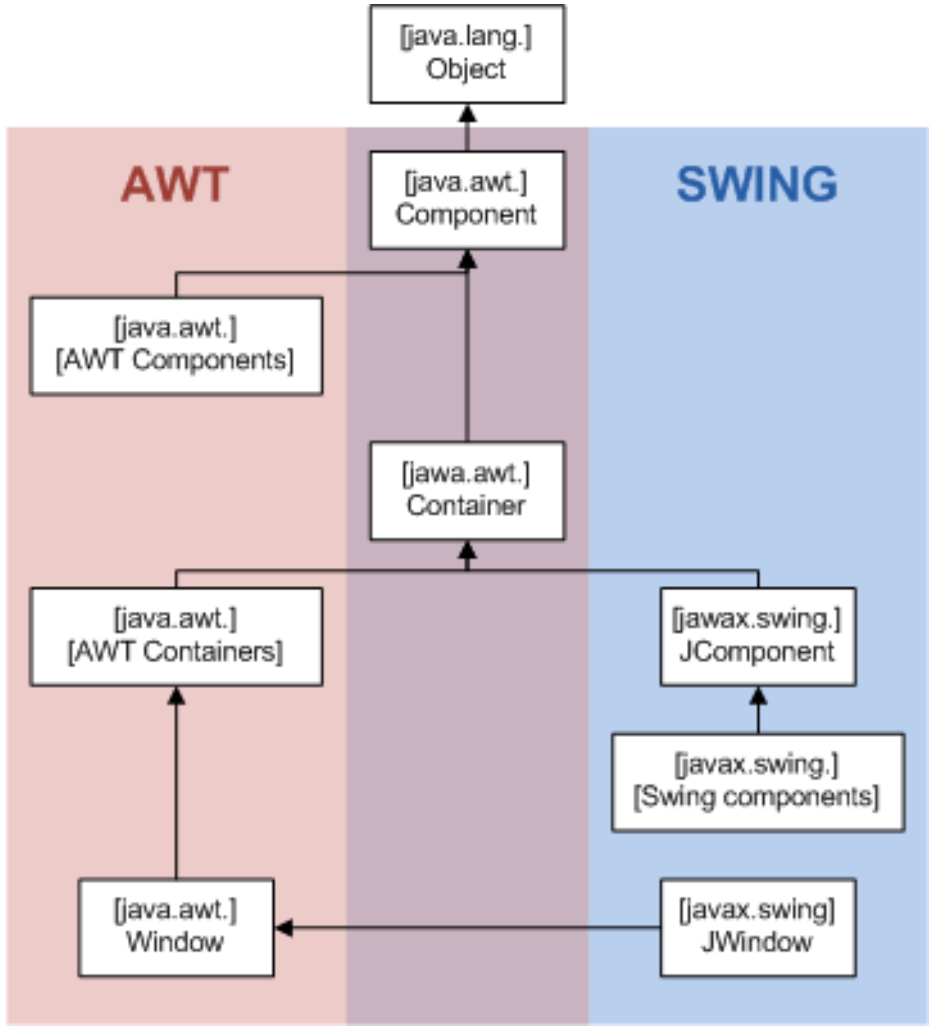
http://www.designinginteractions.com/chapters/4

Graphical User Interfaces in Java

- Abstraction is especially important
  - Displaying graphics is complex
  - Operating system helps, but Java likes to be independent of OS
- Deep hierarchy of interfaces and polymorphism in Java graphics packages

Swing and AWT

- AWT = Abstract Window Toolkit
  `java.awt.*`
- Swing is more modern
  `javax.swing.*`
- Every piece of a Swing GUI is a JComponent



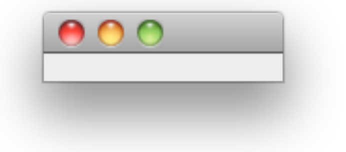
[java.lang.] Object
AWT
[java.awt.] Component
SWING
[java.awt.] [AWT Components]
[jawa.awt.] Container
[java.awt.] [AWT Containers]
[jawax.swing.] JComponent
[javax.swing.] [Swing components]
[java.awt.] Window
[javax.swing] JWindow

JFrame

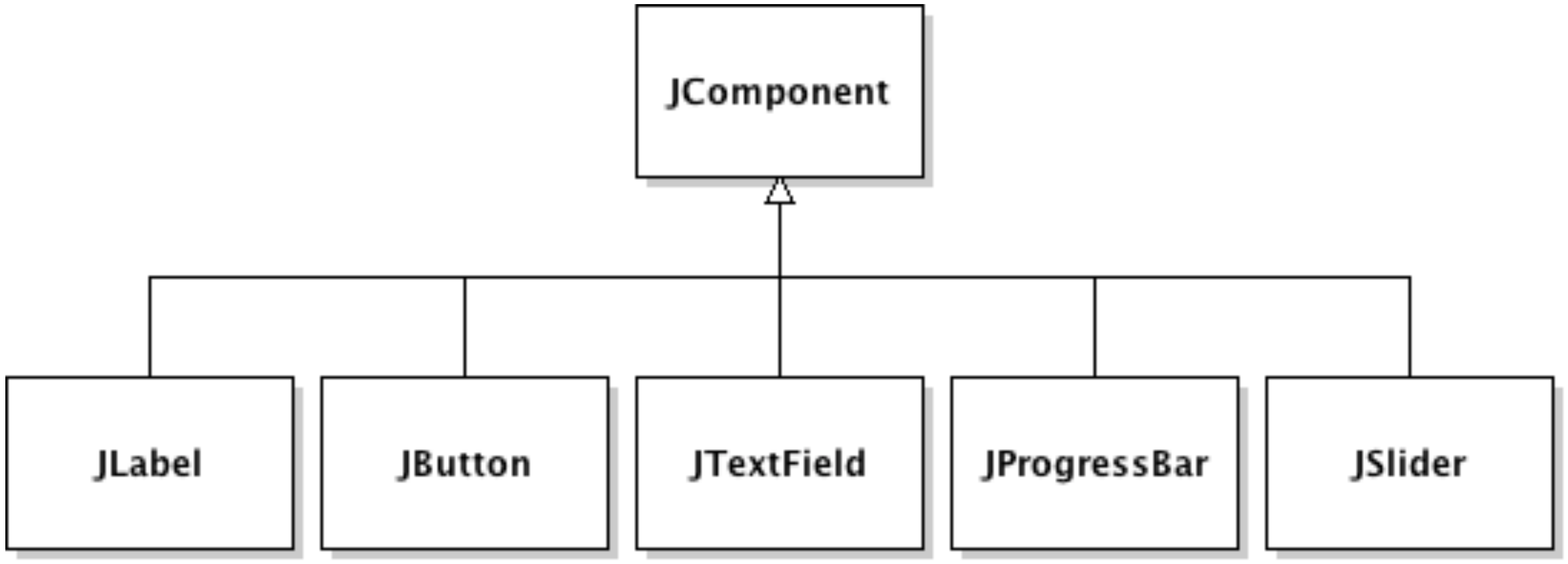
- A JFrame object represents a window
- `void add(Component comp)`
  - adds Component to window
- `void pack()`
  - automatically sizes the window around added Components
- `void setVisible(boolean b)`
  - activates the window

Empty JFrame

```
/**
 * A simple class to experiment with Swing graphics
 * @author bert
 */

import javax.swing.JFrame;

public class GraphicsTester {
   public static void main(String [] args)
   {
      JFrame frame = new JFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.pack();
      frame.setVisible(true);
   }
}
```

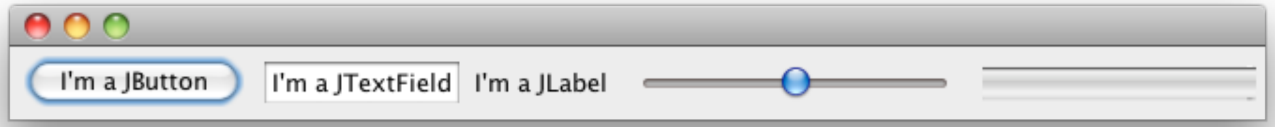


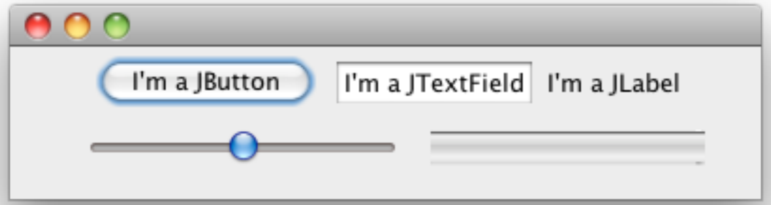
JComponent Subclasses

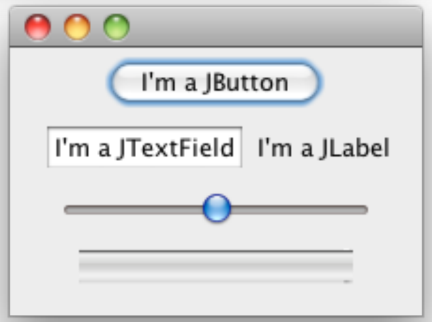


JComponent Subclasses

```
frame.setLayout(new FlowLayout());
frame.add(new JButton("I'm a JButton"));
frame.add(new JTextField("I'm a JTextField"));
frame.add(new JLabel("I'm a JLabel"));
frame.add(new JSlider());
frame.add(new JProgressBar());
```





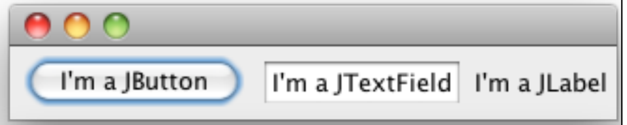


FlowLayout automatically arranges left-to-right as user resizes window

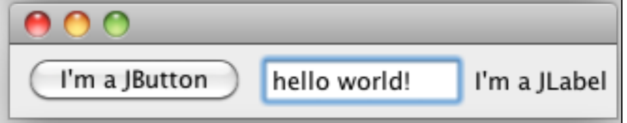
ActionListener Interface

- The `ActionListener` interface provides the method `actionPerformed(ActionEvent e)`
- Each GUI component keeps a collection of `ActionListener` objects
- When the user performs actions on GUI components, each `ActionListener`'s `actionPerformed()` is called

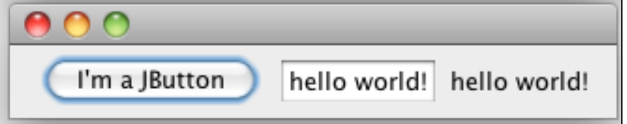
1



2



3



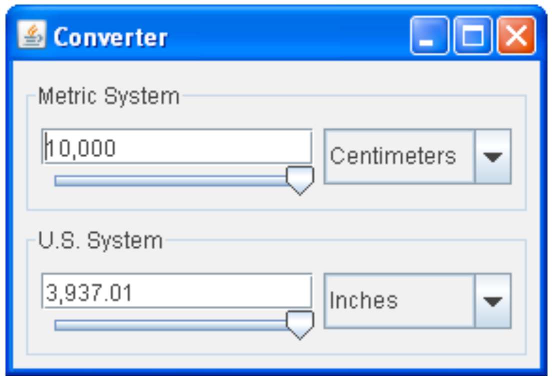
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GraphicsTester2 {
    public static void main(String [] args)
    {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        JButton myButton = new JButton("I'm a JButton");
        final JTextField myTextField = new JTextField("I'm a JTextField");
        final JLabel myLabel = new JLabel("I'm a JLabel");

        myButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent event)
            {
                myLabel.setText(myTextField.getText());
            }
        });

        // continued in box ->
```

```
            //continued
        frame.add(myButton);
        frame.add(myTextField);
        frame.add(myLabel);

        frame.pack();
        frame.setVisible(true);
    }
}
```

Anonymous Classes

- We shorten code via anonymous objects,
  `Scanner s = new Scanner(new File(“input.txt”));`
- Classes that implement interfaces like ActionListener tend to be very specific
- These objects almost always treated as ActionListeners, so true class doesn’t matter
- Define as **anonymous class** inline:
  `ActionListener listener = new ActionListener() { /*...*/ };`

JPanel



- JPanel extends Container and JComponent
- Can be used to hierarchically organize components
- add JPanels to JFrame, add JComponents to your JPanels

image from http://java.sun.com/docs/books/tutorial/uiswing/components/panel.html

Painting Graphics

- All JComponent classes include a method `paint(Graphics g)`
- Swing calls `paint` on the JComponents
- Graphics2D object extends Graphics to include better OOP, rotations, etc.
- Extend JComponent to draw custom GUI elements

Overriding Paint

-
  ```
  public class MyComponent extends JComponent
  {
      /**
       * This method overrides the standard
       * JComponent paint() with our own custom code
       */
      public void paint(Graphics g)
      {
          // Custom drawing code goes here
      }
  }
  ```

Painting to a Graphics2D Object

- `draw(Shape s)`
  - `import java.awt.geom.*; // to use Shape objects`
- `fill(Shape s)`
- `setColor(Color color)`
  - `new Color(int r, int g, int b)`
  - `Color.RED, Color.YELLOW, Color.BLACK, etc`

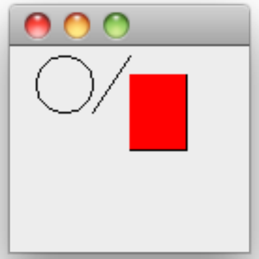
Shape Interface

- Classes that implement Shape: Line2D.Double, Ellipse2D.Double, Rectangle2D.Double
  - Line constructed with `Line(double x1, double y1, double x2, double y2)`
  - Ellipse and Rectangle constructed with (x, y, w, h)

Painting Example

```
import javax.swing.JComponent;
import java.awt.*;
import java.awt.geom.*;

public class MyComponent extends JComponent {

  public MyComponent()
  {
    super();
    this.setPreferredSize(new Dimension(100, 100));
  }

  public void paint(Graphics g)
  {
    Graphics2D g2 = (Graphics2D) g;
    this.setPreferredSize(new Dimension(100,100));

    g2.draw(new Ellipse2D.Double(0, 0, 30, 30));
    g2.draw(new Line2D.Double(50, 0, 30, 30));

    Rectangle2D.Double rect = new
```

```
   public MyComponent()
   {
      super();
      this.setPreferredSize(new Dimension(100, 100));
   }

   public void paint(Graphics g)
   {
      Graphics2D g2 = (Graphics2D) g;
      this.setPreferredSize(new Dimension(100,100));

      g2.draw(new Ellipse2D.Double(0, 0, 30, 30));
      g2.draw(new Line2D.Double(50, 0, 30, 30));

      Rectangle2D.Double rect = new
         Rectangle2D.Double(50, 10, 30, 40);
      g2.draw(rect);

      g2.setColor(Color.RED);

      g2.fill(rect);
   }
}
```



Timer

- Invisible Swing component that can call ActionListeners based on time
- `new Timer(int delay, ActionListener listener)`
- `addActionListener(ActionListener listener)`
- `start()`
- `setRepeats(boolean b) // default true`
- `setDelay(int delay) // milliseconds`

Reading

- This class: Horstmann Ch. 4.6-4.9
  - Try out example code
- Wednesday’s class: Ch. 4.1-4.5, 4.10
- Start/continue Homework 1!