# Event Listeners and Shapes

CSSE 221

Fundamentals of Software Development Honors

Rose-Hulman Institute of Technology

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Announcements

- HW3 solution posted
- Questions on Fifteen or GUIs?
- Please show me your Fifteen UML at your earliest convenience (for example, bring it to Thursday's exam)

# How to do a capsule? Round 2: +Demo and Activity

- I still lecture (15-20 min).
  - You still create a summary and quiz.
- Now, you create the demonstration.
  - Code that shows a concept.
  - How will you know if your classmates are understanding it?
- Now, you create a *hands-on* activity for the class, like?
  - Start the demo code together (like SwingDemo)
  - Have them extend the demo code (like SalariedEmployee)
  - Do a kinesthetic activity (like having the class act out a sort method)
  - Use your creativity!

# More about the Demo/Activity

- Total time for both: ~25 minutes
- **Integrate** your quiz with your demo/activity:
  - 2-3 questions should relate to them.
- Roles of Teammates:
  1. *Demo Driver*: explains the code and adds any live code
  2. *Roving Expert*: checks if any students are having difficulties, asks if they need help
  3. *Questioner*: chooses students to ask the questions on the quiz, asks them, and provides encouragement or corrective feedback as appropriate.

# Capsule Deliverables

- 48 hours in advance:
  - Email me the quiz, key, and summary to me (as before) and a short script of the demo/ activity so I can anticipate what you'll do.
  - **Commit your demo to the csse221-201210-public repo**
    - Include your section number (1 or 2) in the project name
  - I am most available on Monday mornings, Weds and Fridays if you have questions
- Rubric linked to in schedule.

# This week: Fifteen assignment

- Monday:
  - Fifteen specification
  - GUIs using Java's Swing library
  - Intro to UML as a *design* tool
- Tuesday:
  - **EventListeners: responding to user input**
  - **Shape classes**
- Thursday:
  - Anonymous listeners
  - Exam 1

# Exam 1

- Covers through end of week 2 (array lists)
- Thursday evening (~ 2 hours)
- Short written portion: closed-book
- Programming portion: open-book, 221 website (including summaries), Eclipse workspace
  - You may reference any course materials or any code that you did solo or with a partner


ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

## “Fifteen”

- Arrays (especially 2D)
- Creating GUIs using Swing
- Responding to mouse clicks

# Events and listeners

- An *event* is an action taken by the user. For example:
  - Mouse pressed, mouse released, mouse moved, mouse clicked, button clicked, key pressed, menu item selected, slider moved…
- *Event listeners* are code we write that executes when a certain event occurs, taking appropriate action
  - We do this by implementing the corresponding interface.
- We need to add listeners:
  - button.addActionListener(new ClickListener());

Event source

Event responder

# JButton example

2. Responder (this JButton) declares that it *implements ActionListener*

```
public class ExampleButton extends JButton implements ActionListener {
    private ButtonAndMouseFrame frame;
    public ExampleButton(ButtonAndMouseFrame frame) {
        this.frame = frame;
        this.setText("Grow");
        this.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent buttonEvent) {
        this.frame.grow();
    }
}
```

1. JButton says that it will respond to its own button presses

3. Responder (this JButton) implements the required *actionPerformed* method, that says what to do when the JButton is pressed

4. A JButton often refers to one or more other objects (here, the ButtonAndMouseFrame) that it stores in a field. Need a setFrame(frame) method or pass it in the constructor

# Another example: Button in a Panel

- Button is the event source
- Panel has to respond to the event and therefore must listen for events.

```
public TopPanel extends JPanel implements ActionListener {
      private JButton changeColor;
   …
    public TopPanel(){
         this.changeColor = new JButton("Click to change color");
         this.changeColor.addActionListener(this);   //Add the listener to the source
         this.add(changeColor);
    }

  public void actionPerformed(ActionEvent e){
         //Change the background color of the panel
   }
}
```

# Listener interfaces

- **MouseListener**
  - Click, enter, etc.
- **MouseMotionListener**
  - Move and drag
- **ActionListener**
  - Button presses
- **KeyboardListener**
- **ChangeListener**
  - Sliders and where we only care about change
- See the API spec for which methods you need to write

- Question: do I have to write a whole separate class in its own file, just for an actionPerformed method?
- No! You could use an anonymous listener
  - Simpler code, easier access to variables

# Nested classes

- You can define a class inside another class
  - This is called a nested  class
  - It has access to the outer class' fields and methods
  - Useful if the inside class is a "helper class" of interest only to the outside class
- You can define a class and construct an instance of it **inside a method**
  - This is called a local inner class
  - Useful if the class is small and the object refers to variables in the outside class
- You can even make the inside class anonymous.
  - This is called an anonymous  inner class

This nomenclature is not universal.  See http://blogs.oracle.com/darcy/entry/nested_inner_member_and_top for more than you could possibly want to know about this subject

# Back to SwingDemo

- Next stages:
  - Pressing a button changes the panel color
  - Pressing ENTER in textField at the top of the screen changes the panel back to red
- Draw the UML for all classes so far
  - Add the listeners.
    - What other connections do I need?
- Code

# Mouse Adapter class

- OK to leave most of the 5 MouseListener methods empty.
- Alternative is to extend MouseAdapter, then only override the 1-2 you need

# Shapes

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

# The Shape interface

- Methods:
  - **contains()**
  - **intersects()**
  - **getBounds()**
  - **getBounds2D()**
  - **getPathIterator()**

# Who implements Shape?

**new Ellipse2D.Double(double x, double y,**
**double w, double h)**

**new Line2D.Double(Point2D p1, Point2D p2)**

**new Arc2D.Double(double x, double y,**
**double w, double h,**
**double start, double extent,**
**int type)**

- See the javadoc for the Shape interface!
- Point2D.Double does not implement Shape

# Back to Demo

- Create an arbitrary polygon
- If the user moves the mouse within it, then print, "Got me!" to the console.
- Can you do this by yourself?

# Work on Fifteen Spec now

- You need to do 2 things before you start coding:
  - Show us your UML
  - Show us your user stories

# UML ideas

- List of components
- For each component
  - Extends a class?
  - Implements interfaces?
  - Creates instances of other components?
  - Has instances of other components?
- For which objects can I use the default Java version and which do I need to extend?
    - Frames, panels: extend
    - Text boxes: use Java's
    - Buttons: it depends