CSSE 220 Day 8

Event Handling via Listeners

Announcements

- BigRational due now
- Tonight's HW: quick reading (no quiz), finish Swing warmup (2 more short, but non-trivial programs)
- Exam next Friday (28 March); details on Tuesday

- Questions about Java? Reading? Homework?

Some Classes That We will be Using

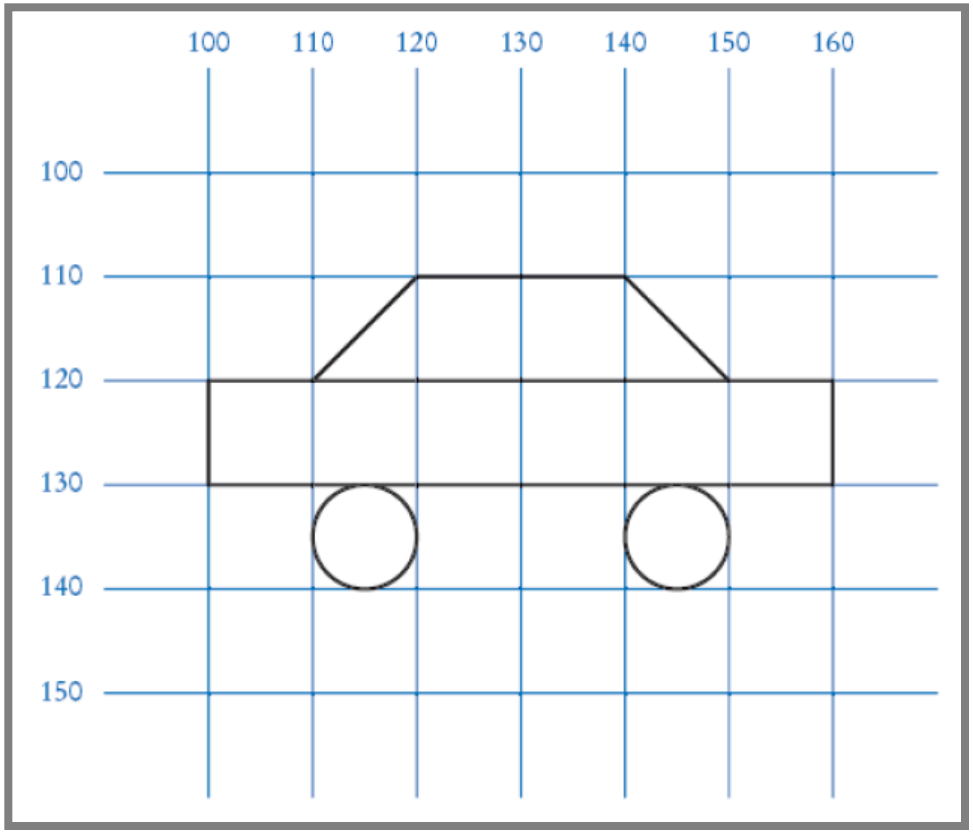
Class	What it is
JFrame	a top-level window
JComponent	a region where we can draw; also parent of many other widget classes
JButton	a JComponent representing a button. When clicked, an action can happen
JLabel	a place to put text in a window
JTextfield	a place for the user to enter text
JPanel	a JComponent that can be used as a container for organizing other widgets
Graphics	an object that can draw things on a JComponent. We never have to create this object; it is provided to us by the system
Graphics2D	a more "object-oriented" graphics object
JOptionPane	Request a single line of input from the user,

GUI recap

- So far we have
  - Created A JFrame to serve as a top level window.
  - Added a subclass of JComponent to the JFrame.
  - Drawn in the component by writing code in the paintComponent() method.
  - Used the Graphics2D object passed to paintComponent by the system.
  - Gotten that object to draw shapes by using Graphics2D's **draw** and **fill** methods.
  - Drawn text and modified colors
  - Constructed colors based on RGB values.

Live Demo – continued

- Look at the Car example.



Event-driven Programming: What?

- So far, we've controlled the flow of our program.
- But modern programs respond to *events*
  - Mouse motion, mouse clicks, button presses, menu selections, …
- The Java window manager generates a huge number of events
  - Whenever any of these happen
- "Most Programs don't want to be flooded by boring events"
  – Cay Horstmann
  - We need to listen for specific events

class Foo implements MouseListener {

…

}

Event-driven Programming: Which?

Interface, that is…

- `ActionListener`
  - For component-defined actions (such as pressing a button)
- `MouseMotionListener`
  - For receiving mouse motion events (movement and dragging) on a component.
- `MouseListener`
  - For clicks and other mouse events (click and double-click, mouse enters component)
- `KeyboardListener`
- `ChangeListener`
  - For components in which we only care about change (like sliders)

See the API spec. for which methods you need to write

Event-driven Programming: How?

- Implement the *Blah*`Listener` interface

```
class Foo implements MouseMotionListener {
  …
  // We promise to implement these.
  void mouseDragged(MouseEvent e) {
    System.out.println("Hey, stop pulling me!");
  }
  void mouseMoved(MouseEvent e) {
    System.out.println("The mouse is moving!");
    System.out.println(e.getX() + " " + e.getY());
  }
}
```

Some demo programs we will write

- ButtonTester/ClickListener
  - About as simple as we can get and still respond to clicks. (from *BigJava*)
  - A separate ActionListener class.
- OneButton
  - Frame is filled with a button that changes colors when clicked.
- FollowTheMouse
  - Draw a small circle where the user clicks.
- OneButton2
  - Make the button smarter …
- ClickCounter
  - Clicking a button causes the contents of a label to change.
  - The Frame is the "boss" and the ActionListener.
- Multiplier
  - Get two numbers from textfields and display their product.

Listeners

- Need 3 things!
  1. Responder implements ActionListener interface
  2. This means it implements actionPerformed method:

```
public void actionPerformed(ActionEvent e) {
    // what happens when button is pressed
}
```

  3. The "listenee" must attach the listener/responder

Say a frame has a button.

```
this.button.addActionListener(this);
```

Listens

Responds (*this* is the frame, but could be a panel or even the button itself)

Example: Button in a Panel

- Just one of many ways to do this…
  - 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
   }
}
```