Swing Widgets

10 Feb 2010
CMPT166
Dr. Sean Ho
Trinity Western University

Outline for today

- Event handling:
  - Types of events
  - Kinds of event handler interfaces
- Swing widgets:
  - Text: JLabel, JTextField, JPasswordField
  - Buttons: JButton
  - JCheckBox, JRadioButton, JComboBox
    - **Use ItemListener interface, itemStateChanged() method, and ItemEvent object**
- See JavaSE API docs or Google “javase (class)”

Event handling

$$\begin{tikzcd}\boxed{\text{widget}} \arrow[r] & \text{event} \arrow[r] & \boxed{\text{listener}}\end{tikzcd}$$

- Window (JFrame) creates widgets in constructor
  - **`JButton quit = new JButton(“Quit”);`**
  - Assigns listeners to each widget
  - **`quit.addActionListener( handler );`**
- Widgets generate Events upon user interaction
  - Or create synthetically, e.g., timers
- Event is passed to corresponding listener
  - **`public void actionPerformed(ActionEvent e)`**
  - Listener acts accordingly
  - Screen is refreshed when listener returns

Which widget fired the event?

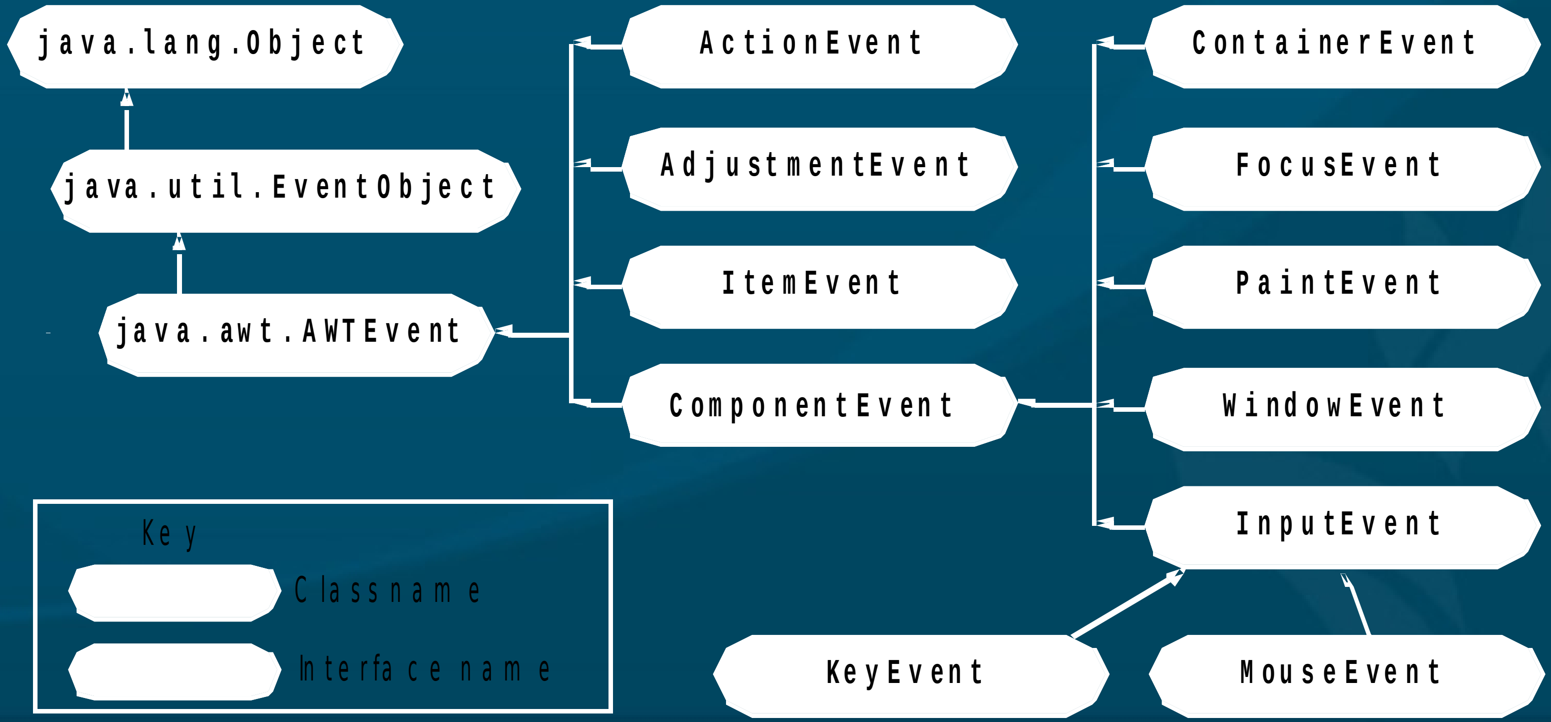
- If all the widgets use the same listener, how can that actionPerformed() method tell which widget generated an event?
  - **public void actionPerformed(ActionEvent e)**
- e.getSource() returns the widget (as an Object)
- e.getActionCommand() returns a string name for the event (default: title of button)
- Can set the action command string directly:
  - **JButton quitButton = new JButton("Quit");**
  - **quitButton.setActionCommand("q");**

Using inner classes as listeners

- Another way: one inner class for each listener
- Each widget uses its own listener object
- Each listener is an instance of its own class
  - public MyWin extends JFrame {
    - public MyWin() {
      - JButton q = new JButton("Quit");
      - q.addActionListener(new QListener());
    - }
    - private class QListener implements ActionListener {
      - public void actionPerformed(ActionEvent e);

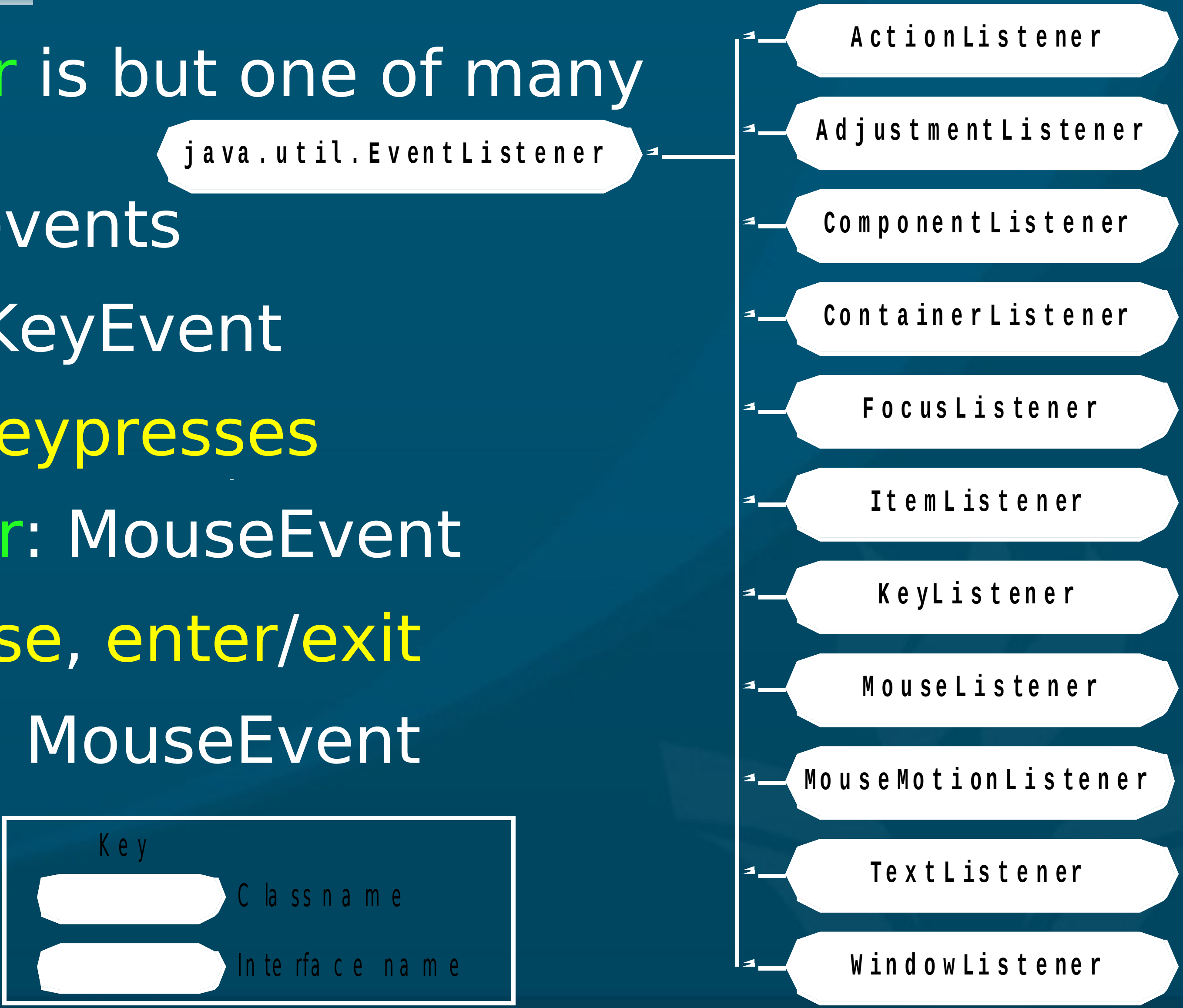
Types of events

- Event classes are in package java.awt.event
- The ActionListener interface uses the actionPerformed() method on an ActionEvent object



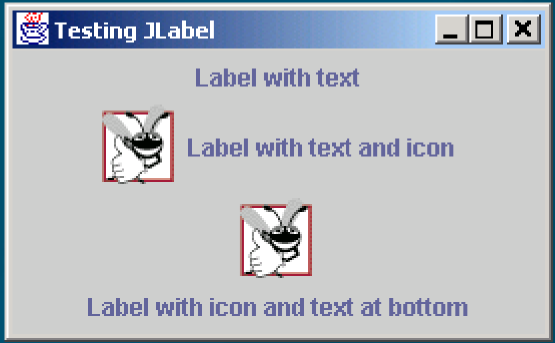
Other EventListener interfaces

- ActionListener is but one of many interfaces for handling events
- KeyListener: KeyEvent
  - Listen for keypresses
- MouseListener: MouseEvent
  - Press/release, enter/exit
- MouseMotion: MouseEvent
  - Move, drag

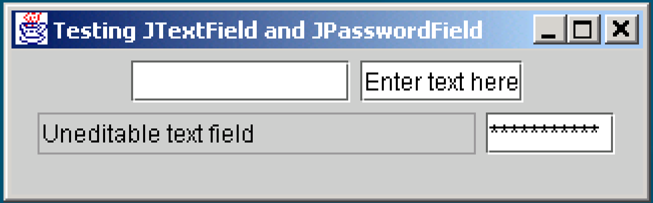


JLabel

- Intended to be a text/image widget describing another component
  - Label1 = new JLabel(“Rotation”)
- Change the text:
  - label1.setText(“Rot”);
- Add a tooltip:
  - label1.setToolTipText(“Rotation in degrees”);
- Add an icon:
  - Icon rotIcon = new ImageIcon(“rot.gif”);
  - label1.setIcon(rotIcon);

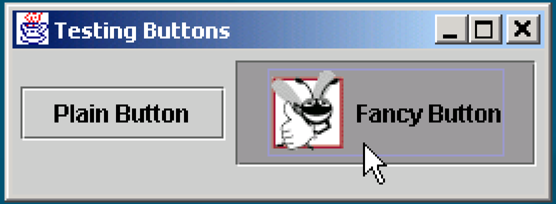


Text fields

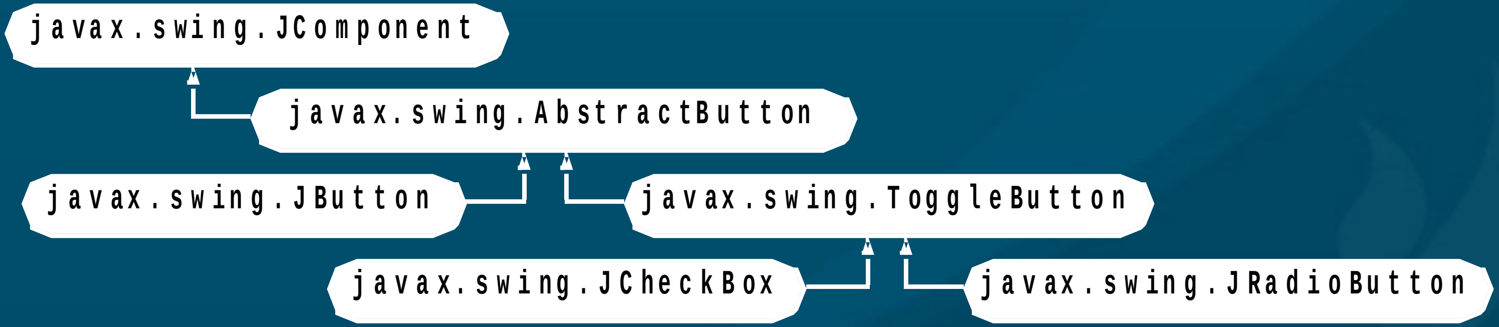


- JTextField:
  - Single-line widget for user to type in text
    - text1 = new JTextField(10); // field width
    - text2 = new JTextField("Type your name here");
    - Read or change the text in the box with .getText() and .setText(String s)
  - Disable user editing:
    - text1.setEditable(false);
- JPasswordField: subclass that shows only dots
- JTextArea: allows multiple lines, word-wrap

JButton



- User clicks to trigger an ActionEvent
- Several types:
  - Command button, check box, toggle, radio
- Abstract superclass: javax.swing.AbstractButton

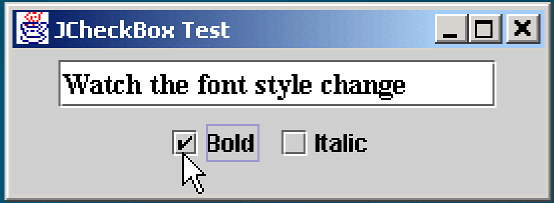


```
Icon rotIcon = new ImageIcon( “rot.png” );
Icon rotIconDown = new ImageIcon( “rotdn.png” );
rotButton = new JButton( “Rotate”, rotIcon );
rotButton.setRolloverIcon( rotIconDown );
```

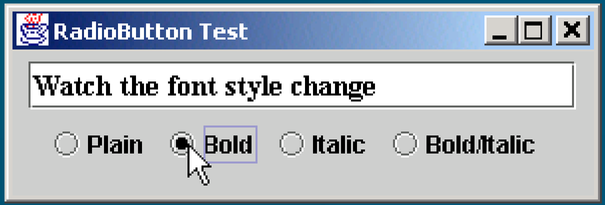
JCheckBox and ItemListener

- JCheckBox uses a different listener interface:
  - **wireframe = new JCheckBox(“Wireframe”);**
  - **MyItemHandler handler = new MyItemHandler();**
  - **wireframe.addItemListener(handler);**
- ItemListener interface uses itemStateChanged() method on an ItemEvent object:
  - **private class MyItemHandler implements ItemListener {**

```
  public void itemStateChanged( ItemEvent event ) {
    if ( event.getSource() == wireframe ) {
      if ( event.getStateChange() ==
         ItemEvent.SELECTED ) {
      …
```



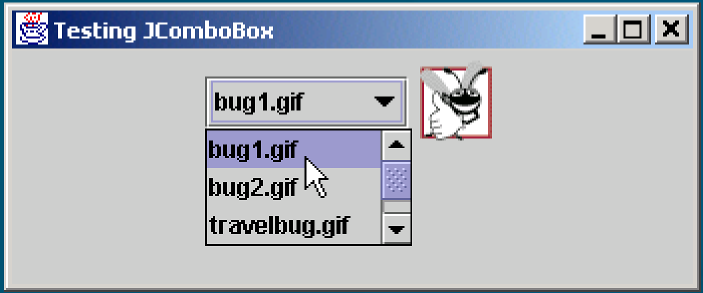
JRadioButton



- triButton = new JRadioButton(“Triangles”, false);
- quadButton = new JRadioButton(“Quads”, true);
- tristripButton = new JRadioButton(“Tristrips”, false);

- Also uses ItemListener:
  - MyItemListener handler = new MyItemListener();
  - triButton.addItemListener(handler);
- Usually put radio buttons into a ButtonGroup:
  - geomGroup = new ButtonGroup();
  - geomGroup.add(triButton);
  - geomGroup.add(quadButton);
  - geomGroup.add(tristripButton);
- This is in addition to add()ing to the window

JComboBox



- Drop-down list for user to choose one entry
  - **private String geom[] = { “Triangles”, “Quads”, “Tristrips” };**
  - **geomCombo = new JComboBox(geom);**
- Show only three rows at a time:
  - **geomCombo.setMaximumRowCount(3);**
- Also uses ItemListener interface
- See which entry user selected (0, 1, 2, etc.):
  - **geomCombo.getSelectedIndex()**