

Ch11: Swing events and widgets

14 March 2007

CMPT167

Dr. Sean Ho

Trinity Western University

Review last time

- More on JOptionPane
- Swing vs. AWT, lightweight vs. heavyweight
- Superclass structure of Swing
- Nested and inner classes
- Event handling
 - Delegate classes

What's on for today

- Event handling
 - Delegate classes
 - Subclasses of `ActionEvent`
 - Sub-interfaces of `EventListener`
- Swing widgets
 - `JLabel`
 - `JTextField`, `JPasswordField`
 - `JButton`
 - `JCheckBox`, `JRadioButton`, `JComboButton`
 - ◆ `ItemListener` interface and `ItemEvent` class

Event handling

- We've seen examples like this:

```
public class Histogram extends JPanel implements
    ActionListener {
    public Histogram() { ...
        widget.addActionListener( this ); ... };    // register
    public void actionPerformed() { ... };
    public static void createAndShowGUI() { ... };
    public static void main() { ... };
}
```

- One class does **three** functions:
 - ◆ `main()/createAndShowGUI()`: setup **window**
 - ◆ **Constructor**: create, layout **widgets**
 - ◆ `actionPerformed()`: **event** handler

Delegate classes

- Alternatively: use separate classes

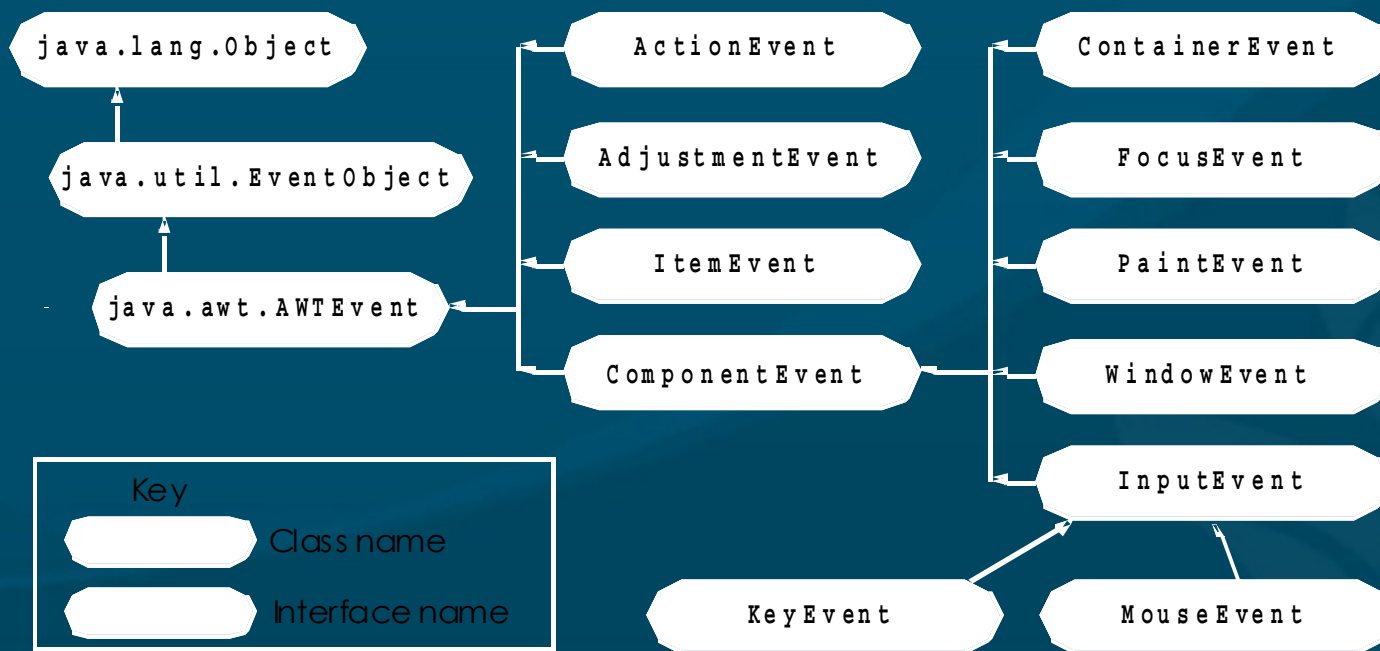
```
public class Histogram extends JPanel {
    public Histogram() { ...
        InputHandler handler = new InputHandler();
        widget.addActionListener( handler ); ... };
    private class InputHandler implements ActionListener {
        public void actionPerformed() { ... };
    }
}

public class HistogramTest { // in separate file
    public static void createAndShowGUI() { ... };
    public static void main() { ... };
}
```

- Uses **inner** class to define event handler

More on event handling

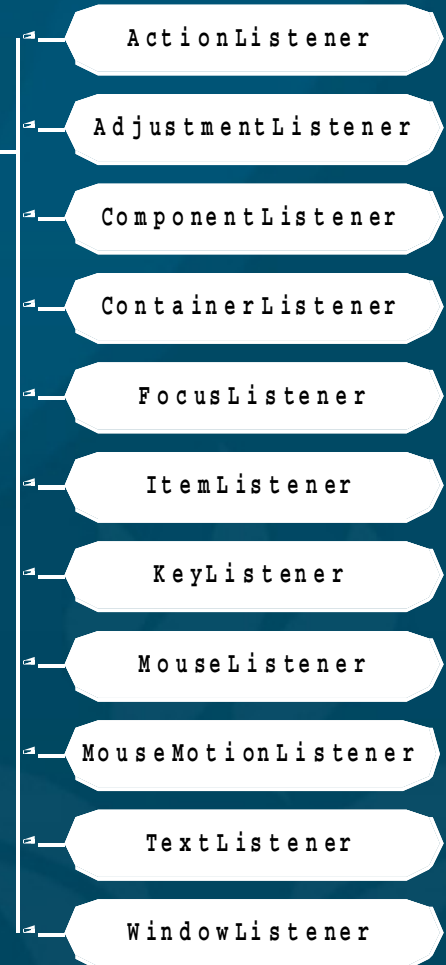
- Event **classes** are in package `java.awt.event`
- The **ActionListener** interface uses the `actionPerformed()` method on an **ActionEvent** object



Other ActionListener interfaces

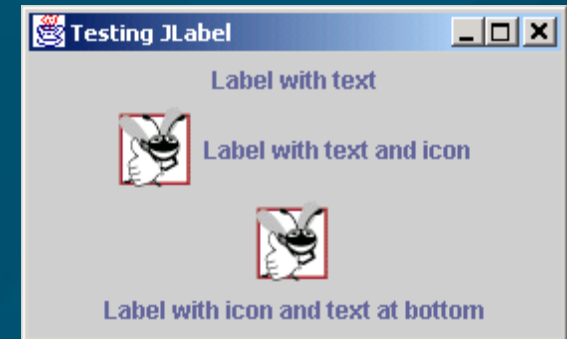
- **ActionListener** is but one of many **interfaces** for handling events
- **KeyListener**: KeyEvent
 - Listen for **keypresses**
- **MouseListener**: MouseEvent
 - **Press/release, enter/exit**
- **MouseMotion**: MouseEvent
 - **Move, drag**

java.util.EventListener



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);`



JTextField and JPasswordField



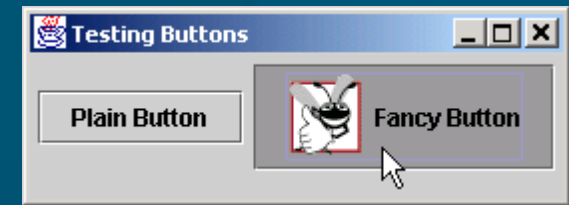
■ JTextField:

- Single-line widget for user to **type in text**
 - ◆ `text1 = new JTextField(10);` // field width
- **Default** text:
 - ◆ `text1 = new JTextField("Type your name here");`
- **Disable** user editing:
 - ◆ `text1.setEditable(false);`

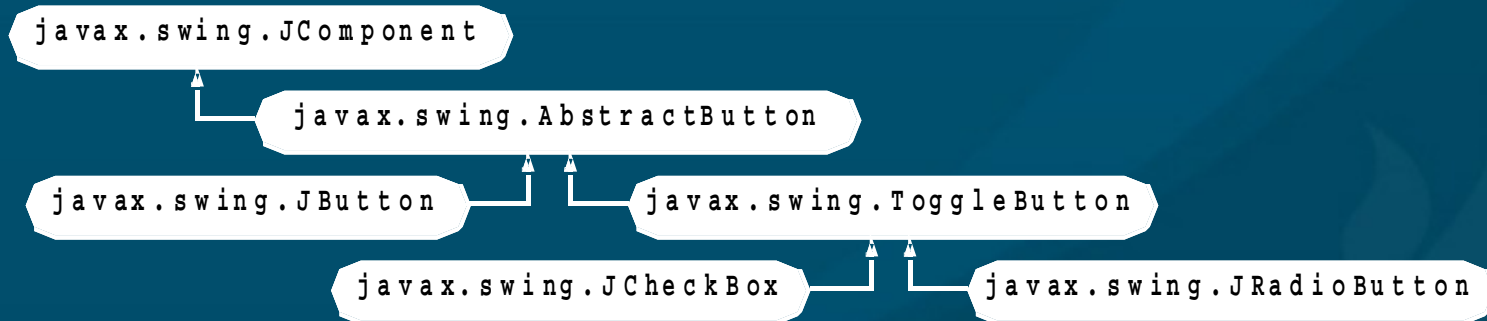
■ JPasswordField: subclass that shows only dots

- The ActionListener event handler should use `event.getActionCommand()` to **get the text**

JButton



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



- ◆ **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:

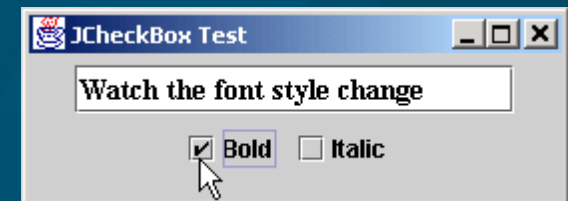
- ◆ `wireframe = new JCheckBox("Wireframe");`
- ◆ `MyItemHandler handler = new MyItemHandler();`
- ◆ `wireframe.addItemListener(handler);`

- ItemListener interface

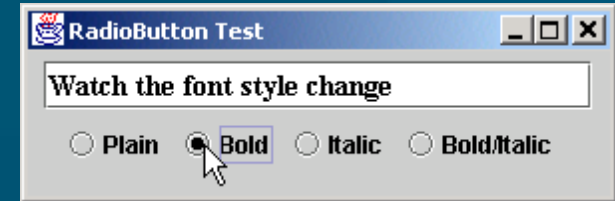
uses `itemStateChanged()` method on an `ItemEvent` object:

- ◆ private class `MyItemHandler` implements `ItemListener` {
 public void `itemStateChanged(ItemEvent event)` {
 if (`event.getSource() == wireframe`) {
 if (`event.getStateChange() == ItemEvent.SELECTED`) {

...



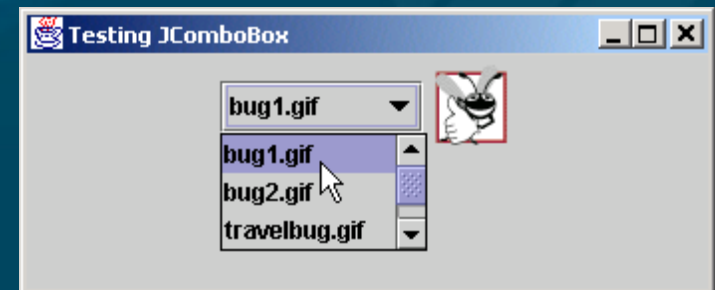
JRadioButton



- ◆ `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 `JPanel`

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()`



Summary of today

- Event handling
 - Delegate classes
 - Subclasses of `ActionEvent`
 - Sub-interfaces of `EventListener`
- Swing widgets
 - `JLabel`
 - `JTextField`, `JPasswordField`
 - `JButton`
 - `JCheckBox`, `JRadioButton`, `JComboButton`
 - ◆ `ItemListener` interface and `ItemEvent` class

TODO

- Lab4 due tonight
 - OO concepts (sets and vectors)