

# CS 251

## Intermediate Programming

### GUIs: Event Listeners

Brooke Chenoweth

University of New Mexico

Spring 2014

# What is an Event Listener?

- A small class that implements a particular listener interface.
- Listener object can register as a listener for events from event source object.
  - One listener object can listen for events from multiple sources.
  - Multiple listeners can listen for events from the same source.

# ActionListener on a Button

## Implement ActionListener interface

---

```
public class MyActionListener
    implements ActionListener {
    public void actionPerformed(ActionEvent ev) {
        System.out.println("Action Happened!");
    }
}
```

and register the listener with a component.

---

```
MyActionListener listener = new MyActionListener();
JButton button = new JButton("My Button");
button.addActionListener(listener);
```

# Anonymous ActionListener

For one-off listeners, we often use an anonymous class.

---

```
JButton button = new JButton("My Button");
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ev) {
        System.out.println("Action Happened!");
    }
});
```

# Common component listeners

**component listener** changes in the component's size, position, or visibility.

**focus listener** whether the component gained or lost the keyboard focus.

**key listener** key presses

**mouse listener** mouse clicks, mouse presses, mouse releases and mouse movement into or out of the component's drawing area.

**mouse-motion listener** changes in the mouse cursor's position over the component.

**mouse-wheel listener** mouse wheel movement over the component.

# Event Listeners

- Look at component to see what sort of listeners it can take.
- Implement listener interface and add to component.
- Listener interfaces with multiple methods generally have an *adapter* class that provides empty implementation.

# Action Listeners

- Most common event handler to implement.
- Action events happen when user performs an action
  - Click a button
  - Select a menu item
  - Press Enter in a text field
  - etc.
- `actionPerformed` message sent to all action listeners registered with the relevant component.

# Click Counter

```
public class ClickCounter extends JFrame
    implements ActionListener {
    private JTextField text = new JTextField(20);
    private JButton button = new JButton("Click me!");
    private int numClicks = 0;

    public ClickCounter() {
        super("Click Counter Demo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        button.addActionListener(this);

        getContentPane().add(button, BorderLayout.CENTER);
        getContentPane().add(text, BorderLayout.PAGE_START);
        pack();
    }

    public void actionPerformed(ActionEvent e) {
        numClicks++;
        text.setText("Button Clicked " + numClicks + " times");
    }
}
```



## Key Listener

- Key events indicate when the user is typing.
- Events fired by component with *keyboard focus*
  - `keyTyped` User typed a Unicode character
  - `keyPressed` User pressed a key
  - `keyReleased` User releases a key
- `KeyEvent` has methods to tell which key and which modifiers (holding down Ctrl or Shift, for example)

# Key Listener Example

```
public class KeyDemo extends JFrame {
    public KeyDemo() {
        super("Key Listener Demo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JTextArea area = new JTextArea(10, 20);
        area.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent ev) {
                System.out.println("Pressed: " +
                    KeyEvent.getKeyText(ev.getKeyCode()));
            }
        });
        getContentPane().add(area);
        pack();
    }

    public static void main(String[] args) {
        new KeyDemo().setVisible(true);
    }
}
```

# Listening to the Mouse

There are three kinds of listeners for the mouse.

- `MouseListener` clicked, pressed, released, entered, exited
- `MouseMotionListener` dragged, moved
- `MouseWheelListener` mouse wheel moved

The `MouseAdapter` class implements them all.

# Mouse Listener Example

```
public class MouseDemo extends JFrame {
    public MouseDemo() {
        super("Mouse Listener Demo");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel();
        panel.setPreferredSize(new Dimension(300,300));
        panel.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ev) {
                System.out.println("Click " + ev.getButton() +
                                   " at: " + ev.getPoint());
            }
        });
        getContentPane().add(panel);
        pack();
    }

    public static void main(String[] args) {
        new MouseDemo().setVisible(true);
    }
}
```

# Threads in GUI

- Initial thread: `main`
- Event dispatch thread: handles GUI
- Worker threads: larger tasks in the background.

# Start GUI on event dispatch thread

Previous examples have been living dangerously, creating GUI on main thread.

---

```
public static void main(String[] args) {  
    createAndShowGUI();  
}
```

Really should use be creating GUI on event dispatch thread.

---

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            createAndShowGUI();  
        }  
    });  
}
```