Graphical User Interface

Hirra Anwar

Agenda

- **Event Driven Programming**
- **Java Event Types and Listeners Interfaces**
  - Action Listener
  - Mouse Listeners
  - Key Events
  - ……
- **Adapter Classes**
- **Anonymous Inner Classes**

Agenda

- **Event Driven Programming**
- **Java Event Types and Listeners Interfaces**
  - Action Listener
  - Mouse Listeners
  - Key Events
  - ……
- **Adapter Classes**
- **Anonymous Inner Classes**

- **Event Driven Programming**
- **Java Event Types and Listeners Interfaces**
  - Action Listener
  - Mouse Listeners
  - Key Events
  - ……
- **Adapter Classes**
- **Anonymous Inner Classes**

- **The Delegation Event Model**
  - Defines standard and consistent mechanisms to generate and process events
  - Describes how your program can respond to user interaction
  - Model used by Java to handle user interaction with GUI components

- A source generates an event and sends it to one or more listeners.
- In this scheme, the listener simply waits until it receives an event. Once received, the listener processes the event and then returns.

Three important players

1. Event Source
2. Event Listener/Handler
3. Event Object

- **Event Source**
  - GUI component that generates the event
  - Example: button
- **Event Listener/Handler**
  - Receives and handles events
  - Contains business logic
  - Example: displaying information useful to the user, computing a value
- **Event Object**
  - Created when an event occurs (i.e., user interacts with a GUI component)
  - Contains all necessary information
  - Represented by an Event class

- **Event Driven Programming**
- **Java Event Types and Listeners Interfaces**
  - Action Listener
  - Mouse Listeners
  - Key Events
  - ……
- **Adapter Classes**
- **Anonymous Inner Classes**

- All Events are objects of Event Classes.
- All Event Classes are derived from EventObject.
- When an Event occurs, Java sends a message to all registered Event Listeners from the Event source

- https://docs.oracle.com/javase/7/docs/api/javax/swing/event/package-tree.html

- **The *EventObject class***
  - Found in the *java.util package*
- **The *AWTEvent class***
  - An immediate subclass of *EventObject*
  - Defined in *java.awt package*
  - Root of all AWT-based events

1. Either **implements** a listener interface or extends **a** class that implements a listener interface

2. Register your listener

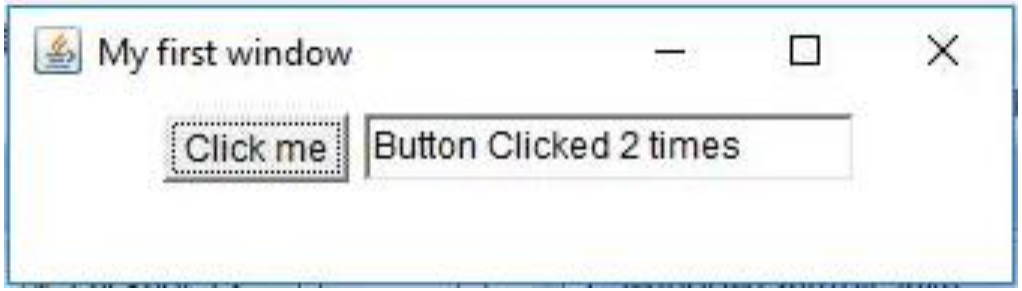
   someComponent.addActionListener(instanceOfMyClass);

3. Implement user action

   public void actionPerformed(ActionEvent e) {
   ...//code

   that reacts to the action... }

Event : Example



```
import java.awt.*;
import java.awt.event.*;

public class AL extends Frame implements ActionListener {
        TextField text = new TextField(20);
        Button b;
        private int numClicks = 0;

public AL(String title) {

                super(title);
                setLayout(new FlowLayout());
                b = new Button("Click me");
                add(b);
                add(text);
                b.addActionListener(this);
        }
```

Event : Example

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

public static void main(String[] args) {
        AL myWindow = new AL("My first window");
        myWindow.setSize(350,100);
        myWindow.setVisible(true);
    }
}
```