Event Handling

GUI are Event Driven

Events

- GUIs generate events when the user interacts with GUI
- For example,
  - Clicking a button
  - Moving the mouse
  - Closing Window etc
- In java, events are represented by Objects
  - These objects tells us about event and its source. Examples are
    - ActionEvent (Clicking a button)
    - WindowEvent (Doing something with window e.g. closing, minimizing)
- Both AWT and swing components (not all) generate events
  - java.awt.event.*
  - java.swing.event.*

Some event classes of java.awt.event

Event Handling Model

- Common for both AWT and Swing components
- Event Delegation Model
  - Processing of an event is delegated to a particular object (handlers) in the program
  - Publish-Subscribe
  - Separate UI code from program logic

Event Handling Steps

- For a programmer the event Handling is a three step process in terms of code
  - Step 1
    - Create components which can generate events
  - Step 2
    - Build component (objects) that can handle events (Event Handlers)
  - Step 3
    - Register handlers with generators
Event Handling Process [1]
Event Generators
- You have already seen a lot of event generators
  • Buttons
  • Mouse
  • Key
  • Window
  Etc
- JButton b1 = new JButton("Hello");
- Now b1 can generate events

Event Handling Process [2]
Event Handlers/Event Listener
• First Technique - By Implementing Listener Interfaces
  • Java defines interfaces for every event type
  • If a class needs to handle an event, it needs to implement the corresponding listener interface
  • To handle "ActionEvent" a class needs to implement "ActionListener"
  • To handle "KeyEvent" a class needs to implement "KeyListener"
  • To handle "MouseEvent" a class needs to implement "MouseListener"
  And so on

Event Listener interfaces of package java.awt.event
<table>
<thead>
<tr>
<th>Class name</th>
<th>Interface name</th>
</tr>
</thead>
<tbody>
<tr>
<td>JButton</td>
<td>java.awt.event.ActionEvent</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.ActionListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.ComponentListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.ContainerListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.FocusListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.TextListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.AdjustmentListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.ItemListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.KeyListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.MouseMotionListener</td>
</tr>
<tr>
<td>JButton</td>
<td>java.awt.event.WindowListener</td>
</tr>
</tbody>
</table>

Example Listeners
public interface ActionListener {
  public void actionPerformed(ActionEvent e);
}

public interface ItemListener {
  public void itemStateChanged(ItemEvent e);
}

public interface ComponentListener {
  public void componentHidden(ComponentEvent e);
  public void componentMoved(ComponentEvent e);
  public void componentResized(ComponentEvent e);
  public void componentShown(ComponentEvent e);
}

Event Handling Process [3]
Event Handlers
• By implementing an interface the class agrees to implement all the methods that are present in that interface.
• Implementing an interface is like signing a contract
• Inside the method the class can do whatever it wants to do with that event
• Event Generator and Event Handler can be the same or different classes

Event Handling Process [4]
Event Handlers
• To handle events generated by Button. A class needs to implement ActionListener interface.
• public class Test implements ActionListener{
  public void actionPerformed(ActionEvent ae){
    // do something
  }
}
Event Handling Process [4]
Registering Handler with Generator

- The event generator is told about the object which can handle its events
- Event Generators have a method
  - `add_________Listener(_________)`
- `b1.addActionListener(objectOfTestClass)`

---

Event Handling: Simple Example
Scenario

When Hello button is pressed, the Dialog box would be displayed

---

**Event Handling: Simple Example**

**Step 1 (cont.)**

/* This program demonstrates the handling of Action Event. Whenever “Hello” button is presses, a dialog box would be displayed in response containing some informative message */

```java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class ActionEventTest{
    JFrame frame;
    JButton bHello;
    public void initGUI (){ /*your code*/
        // Event Generator
        bHello = new JButton("Hello");
        Container con = frame.getContentPane();
        con.add(bHello);
        frame.setSize(200,200);
        frame.setVisible(true);
    }
    public void actionPerformed (ActionEvent ae ){
        JOptionPane.showMessageDialog("Hello is pressed");
    }
    public class ActionEventTest implements ActionListener {
        public void actionPerform (ActionEvent ae ){
            JOptionPane.showMessageDialog("Hello is pressed");
        }
    }
}
```

---

**Event Handling: Simple Example (cont.)**

**Step 2**

```java
//import your packages
public class ActionEventTest implements ActionListener {
    ........
    public void initGUI() ....
    public void actionPerformed (ActionEvent ae ){
        JOptionPane.showMessageDialog("Hello is pressed");
    }
}
```
Event Handling: Simple Example
Step 3 (cont.)

public void initGUI () {
    // Event Generator
    bHello = new JButton("Hello");
    Container con = frame.getContentPane();
    con.add(bHello);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(200, 200);
    frame.setVisible(true);
    // Event Registration
    bHello.addActionListener(this);
}
//end initGUI

public ActionEventTest() {
    initGUI();
}

public static void main(String args[]) {
    ActionEventTest aeTest = new ActionEventTest();
}
//end ActionEvent class

Event Handling: Simple Example
Complete Code

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class ActionEventTest implements ActionListener {
    JFrame frame;
    JButton bHello;
    public void initGUI () {
        frame = new JFrame();
        // Event Generator
        bHello = new JButton("Hello");
        Container con = frame.getContentPane();
        con.add(bHello);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 200);
        frame.setVisible(true);
        // Event Registration
        bHello.addActionListener(this);
    }
    // end initGUI
    public void actionPerformed(ActionEvent ae) {
        JOptionPane.showMessageDialog("Hello is pressed");
    }
    public ActionEventTest() {
        initGUI();
    }
    public static void main(String args[]) {
        ActionEventTest aeTest = new ActionEventTest();
    }
//end ActionEvent class

Behind the Scenes

Event Handling Participants

1. Event Generator / Source
   - Swing and awt components
   - For example, JButton, JTextField, JFrame etc
   - Generates an event object
   - Registers listeners with itself

2. Event Object
   - Encapsulate information about event that occurred
     and the source of that event
   - For example, if you click a button, ActionEvent object is created

Event Handling Participants (cont.)

3. Event Listener/handler
   - Receives event objects when notified, then responds
   - Each event source can have multiple listeners registered on it
   - Conversely, a single listener can register with multiple event
     sources

©Umair Javed CS
Event Handling Participants (cont.)

4. JVM
- Receives an event whenever one is generated
- Looks for the listener/handler of that event
- If exist, delegate it for processing
- If not, discard it (event).

Event Handling Diagram

Making Small Calculator Example Code

Code: Small Calculator

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class SmallCalcApp implements ActionListener {
    JFrame frame;
    JLabel firstOperand, secondOperand, answer;
    JTextField op1, op2, ans;
    JButton plus, mul;
    public void initGUI() {
        frame = new JFrame("Small Calculator");
        frame.setSize(400, 200);
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        firstOperand = new JLabel("First Operand");
        secondOperand = new JLabel("Second Operand");
        answer = new JLabel("Answer");
        op1 = new JTextField(20);
        op2 = new JTextField(20);
        ans = new JTextField(20);
        plus = new JButton('+');
        mul = new JButton('*');
        plus.addActionListener(this);
        mul.addActionListener(this);
        frame.add(firstOperand);
        frame.add(op1);
        frame.add(secondOperand);
        frame.add(op2);
        frame.add(answer);
        frame.add(plus);
        frame.add(mul);
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent event) {
        String oper, result;
        int num1, num2, res;
        oper = op1.getText();
        num1 = Integer.parseInt(oper);
        oper = op2.getText();
        num2 = Integer.parseInt(oper);
        res = num1 + num2;
        result = res + "";
        ans.setText(result);
    }
    
    public static void main(String[] args) {
        new SmallCalcApp();
    }
}
```

Code: Small Calculator (cont.)

```
// providing definition of interface ActionListner's methos
public void actionPerformed(ActionEvent event) {
    String oper, result;
    int num1, num2, res;
    oper = event.getActionCommand();
    num1 = Integer.parseInt(oper);
    oper = event.getActionCommand();
    num2 = Integer.parseInt(oper);
    res = num1 + num2;
    result = res + "";
    ans.setText(result);
}
```
else if (event.getSource() == mul) {
    oper = op1.getText();
    num1 = Integer.parseInt(oper);
    oper = op2.getText();
    num2 = Integer.parseInt(oper);
    res = num1*num2;
    result = res+"";
    ans.setText(result);
}

} // end actionPerformed method

/*write default constructor and call initGUI*/

public static void main (String args[]) {
    SmallCalcApp scApp = new SmallCalcApp();
}

} // end SmallCalcApp