FAST

National University of Computer and Emerging Sciences Peshawar

Lecture # 20

Software Construction and Development (Java Programming)

Instructor: Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE

الذی علم بالقلم۔ علم الانسان ما لم یعلم۔

Event Handling in Java

Contents

Event Handling

One of the most important aspects of most non-trivial applications (especially UI type-apps) is the ability to respond to events that are generated by the various components of the application, both in response to user interactions and other system components such as client-server processing.

GUIs generate events when the user interacts with GUI. For example,

- Clicking a button
- Moving the mouse
- Closing Window etc.

Event Handling…

Both AWT and swing components (not all) generate events

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

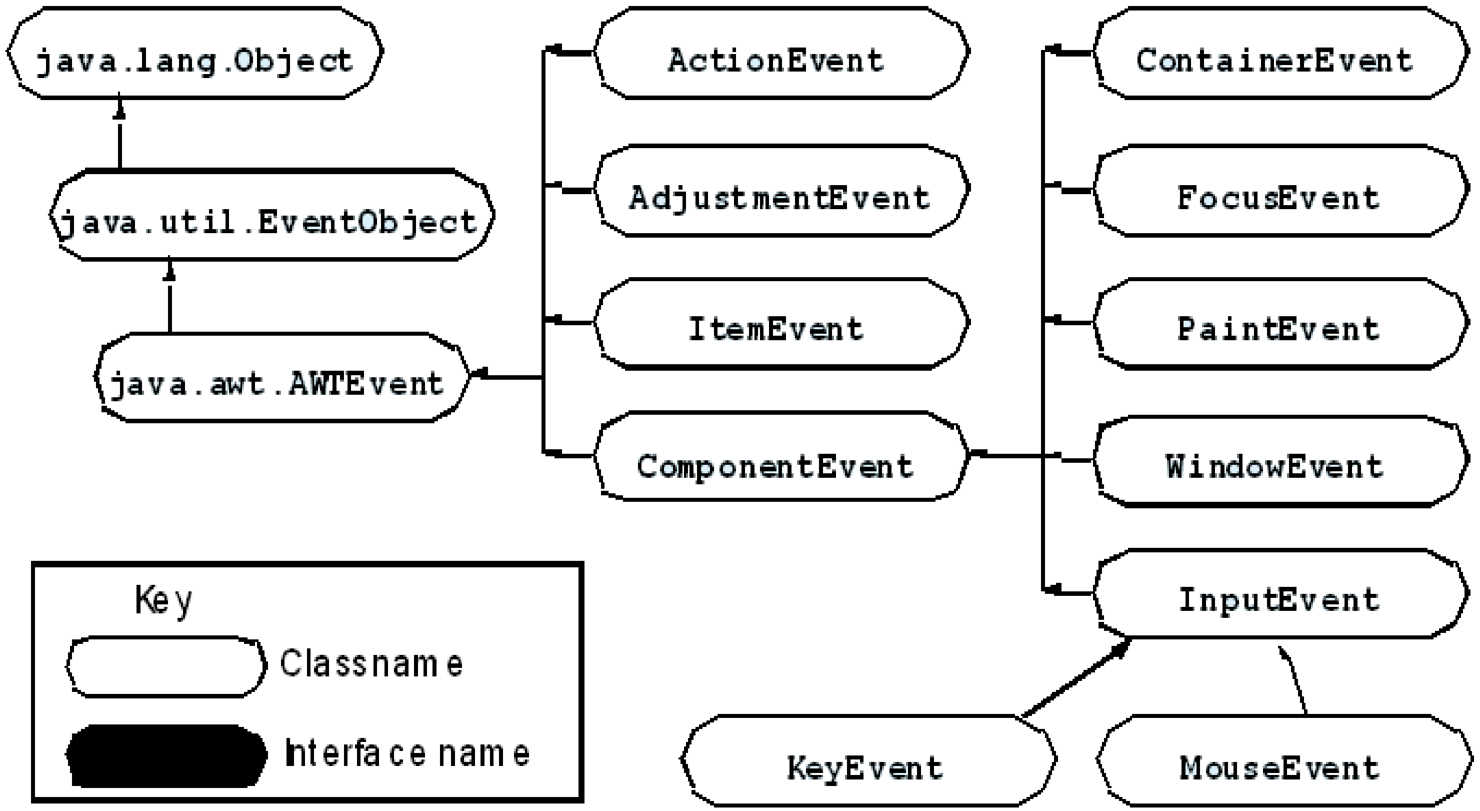
In java, events are represented by Objects

These objects tell us about event and its source. Examples are:

- ActionEvent (Clicking a button)
- WindowEvent (Doing something with window e.g. closing , minimizing)

Event Handling...

Some event classes of java.awt.event are shown in diagram below:



Event Handling Model

- In Java both AWT and Swing components use Event Delegation Model.
- In this model processing of an event is delegated to a particular object (handlers) in the program
- It’s a Publish-Subscribe model. That is, event generating component publish an event and event handling components subscribe for that event. The publisher sends these events to subscribers. Similar to the way that you subscribe for newspaper and you get the newspaper at your home from the publisher.

Event Handling Model…

- This model separates UI code from program logic, it means that we can create separate classes for UI components and event handlers and hence business/program logic is separated from GUI components.

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 (Event Generators)
- **Step 2:** Build component (objects) that can handle events (Event Handlers)
- **Step 3:** Register handlers with generators

Event Handling Process

Step 1: Event Generators

The first step is that you create an event generator. You have already seen a lot of event generators like:

- Buttons
- Mouse
- Key
- Window etc.

Event Handling Process…

Most of GUI components can be created by calling their constructors. For example

JButton b1 = new JButton(“Hello”);

Now b1 can generate events

Event Handling Process...

Step 2: Event Handlers/ Event Listener

The second step is that you build components that can handle events

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".

Event Handling Process...

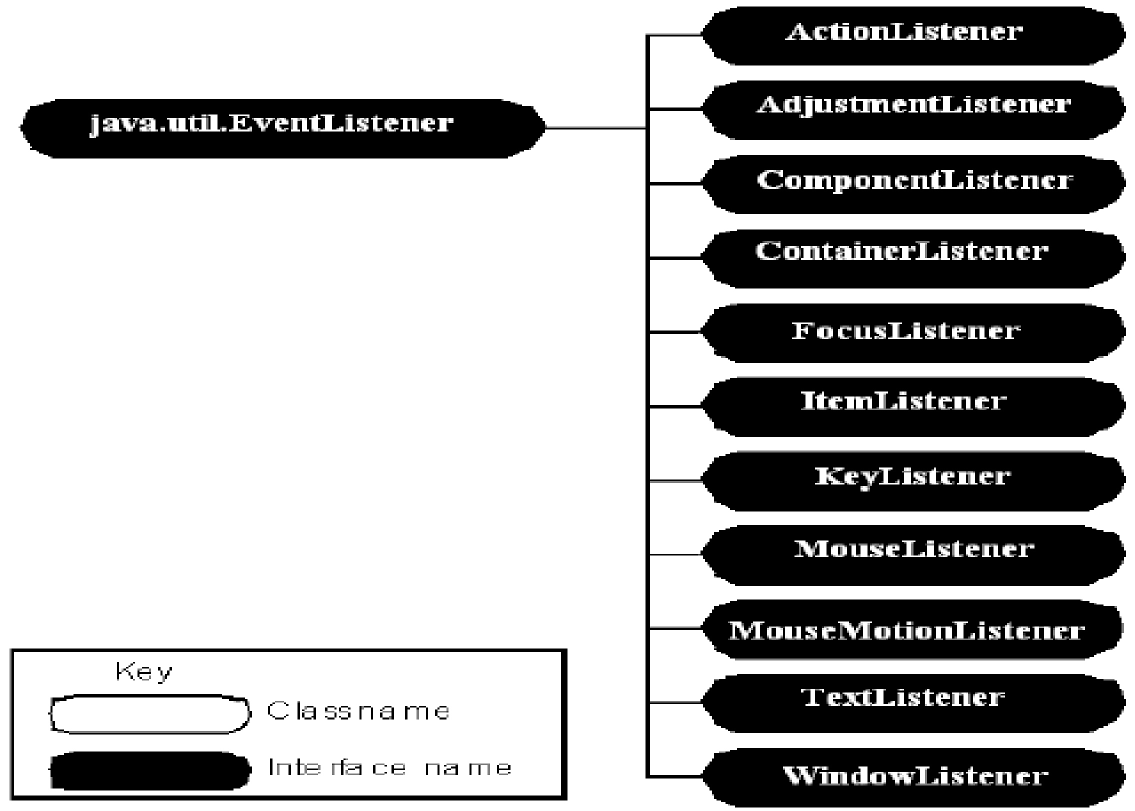
Step 2: Event Handlers/ Event Listener...

- To handle “KeyEvent” a class needs to implement “KeyListener”.

- To handle “MouseEvent” a class needs to implement “MouseListener” and so on.

- Package java.awt.event contains different event Listener Interfaces which are shown in the following figure

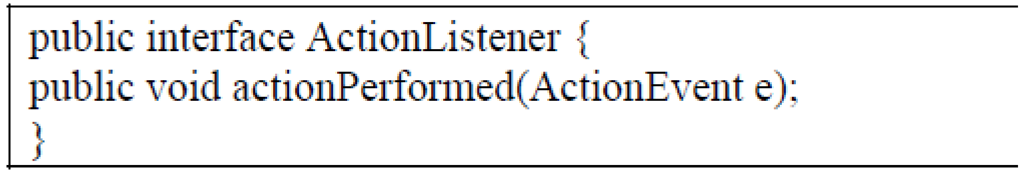
Event Handling Process…



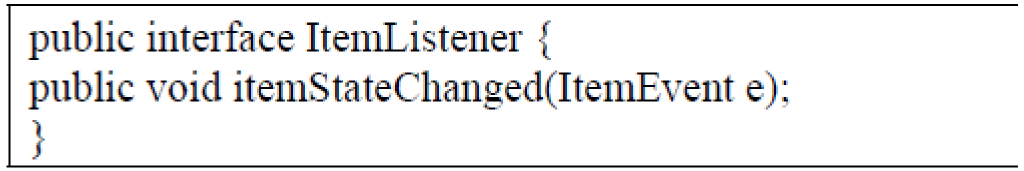
Event Handling Process…

Step 2: Event Handlers/ Event Listener...

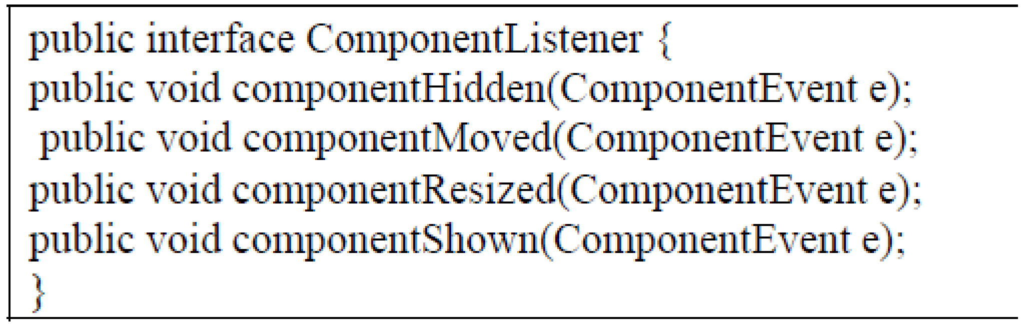
Some Example Listeners, the way they are defined in JDK by Sun



```
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...

Step 2: Event Handlers/ Event Listener...

- 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...

Step 2: Event Handlers/ Event Listener...

- To handle events generated by Button. A class needs to implement ***ActionListener*** interface and thus needs to provide the definition of ***actionPerformed()*** method which is present in this interface.

Event Handling Process…

```
public class Test implements ActionListener{
public void actionPerformed(ActionEvent ae)
{
// do something
}
}
```

Event Handling Process…

Step 3: Registering Handler with Generator

- The event generator is told about the object which can handle its events
- Event Generators have a method

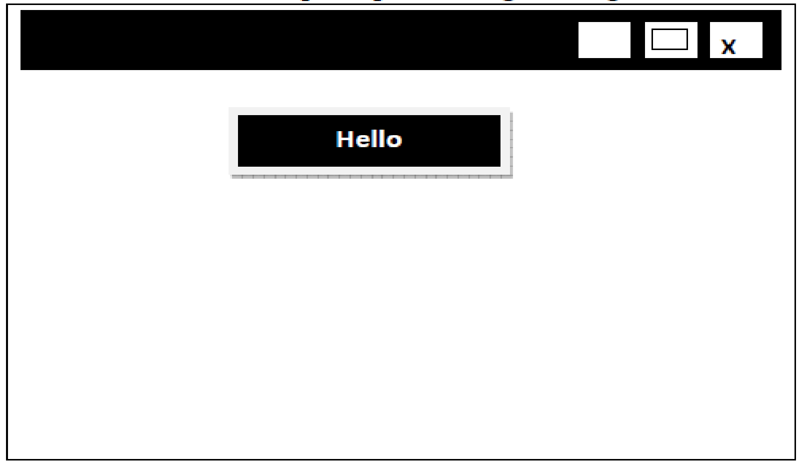
    addXXXListener(_reference to the object of Handler class_)

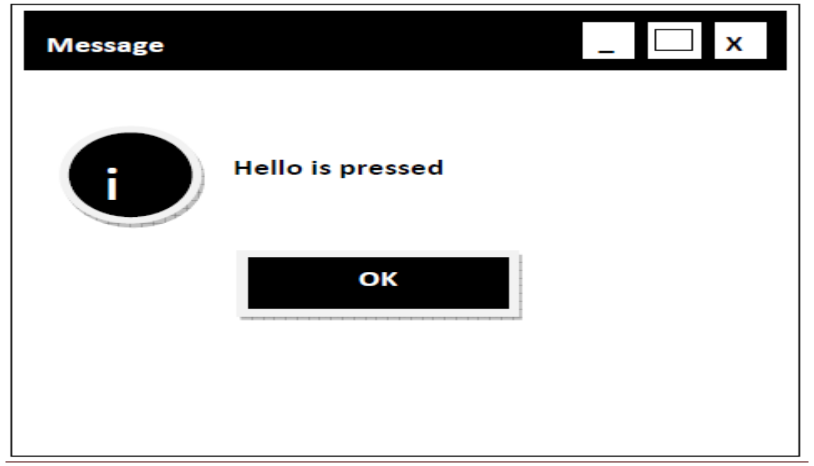
- For example, if b1 is JButton then

    b1.addActionListener(this); // if listener and generator are same class

Event Handling Example

Clicking the “Hello” button will open up a message dialog shown below.





Hello is pressed

We will take the simplest approach of creating handler and generator in a single class. Button is our event generator and to handle that event our class needs to implement ActionListener Interface and to override its actionPerformed method and in last to do the registration.

Event Handling Example

```
1. import java.awt.*;

2. import javax.swing.*;

3. import java.awt.event.*;

/* Implementing the interface according to the type of the event, i.e. creating
event handler (first part of step 2 of our process)

*/

4. public class ActionEventTest implements ActionListener{

5. JFrame frame;

6. JButton hello;
```

Event Handling Example…

```
// setting layout components

7. public void initGUI ( ) {

8. frame = new JFrame();

9. Container cont = frame.getContentPane();

10. cont.setLayout(new FlowLayout());

//Creating event generator step-1 of our process

11. hello = new JButton("Hello");

/* Registering event handler with event generator. Since event handler is in same object
that contains button, we have used this to pass the reference.(step 3 of the process) */

12. hello.addActionListener(this);
```

Event Handling Example…

```
cont.add(hello);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(150, 150);
frame.setVisible(true);
}  // initGUI ( ) method body closed
//constructor
public ActionEventTest ( ) {
initGUI();
}
```

Event Handling Example…

```
/* Override actionPerformed method of ActionListener’s interfacemethod of which will
be called when event takes place (second part of step 2 of our process) */
public void actionPerformed(ActionEvent event) {
JOptionPane.showMessageDialog(null,"Hello is pressed");
}
public static void main(String args[]) {
ActionEventTest aeTest = new ActionEventTest();
}
}       // end class
```

How Event Handling Participants Interact Behind the Scenes?

We have already seen that what a programmer needs to do handle events. Let's see what takes place behind the scenes, i.e How JVM handles event. Before doing that lets revisit different participants of Event Handling Process and briefly what they do.

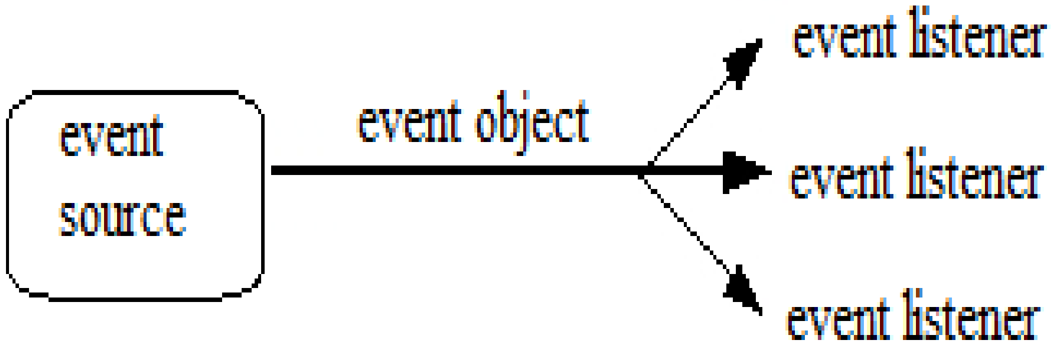
1. Event Generator / Source

- Swing and awt components
- For example, JButton, JTextField, JFrame etc
- Generates an event object
- Registers listeners with itself

How Event Handling Participants Interact Behind the Scenes?...

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.



How Event Handling Participants Interact Behind the Scenes?...

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.

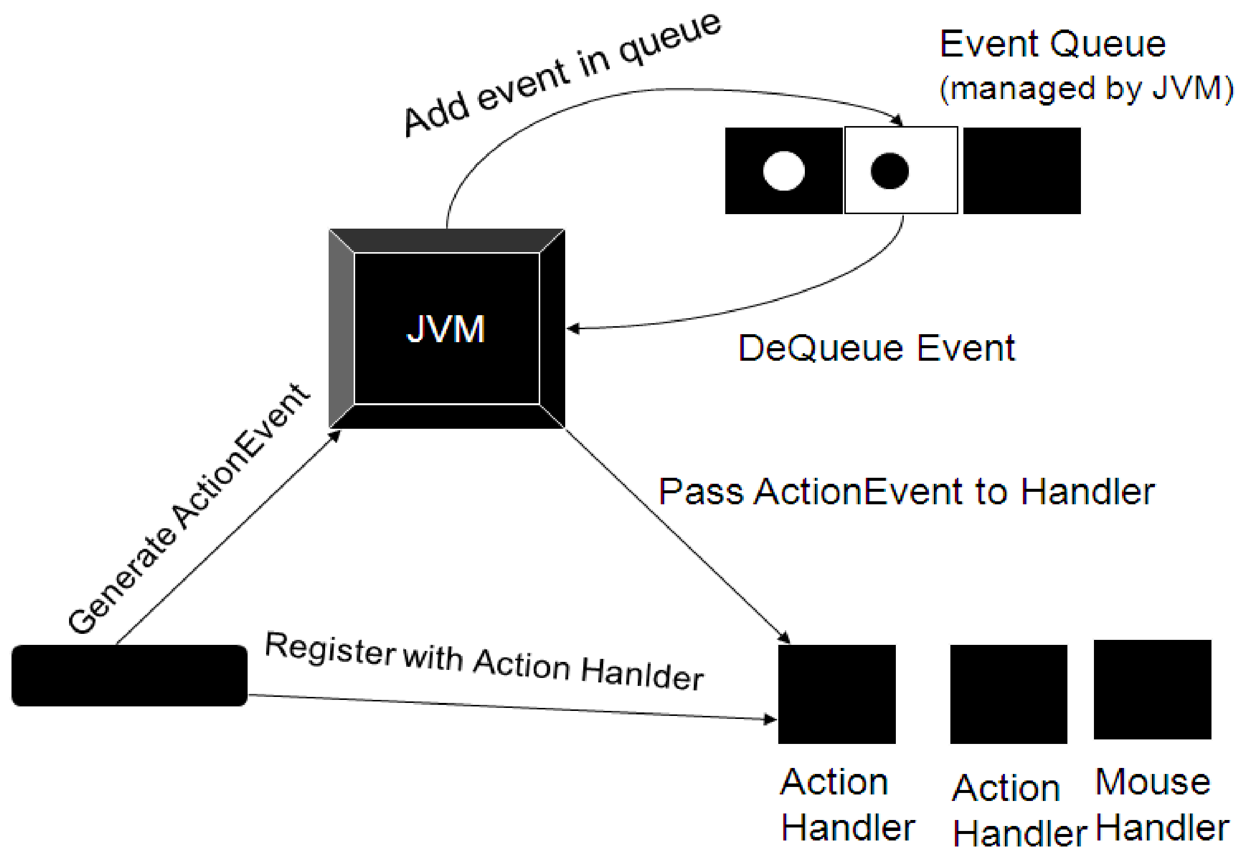
How Event Handling Participants Interact Behind the Scenes?...

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).

When button generates an ActionEvent it is sent to JVM which puts it in an event queue. After that when JVM find it appropriate it de-queue the event object and send it to all the listeners that are registered with that button. This is all what we shown in the pictorial form below:

How Event Handling Participants Interact Behind the Scenes?...



Tasks

1. Write a java program that will change background color of Frame(Container).
2. Write a java program to take your first and last name from user and show it on frame using JLabel.

THANK YOU