Event Handling

Event Handling

What is an Event?

- Change in the state of an object is known as Event, i.e., event describes the change in the state of the source.
- Events are generated as a result of user interaction with the graphical user interface components.
- For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list, and scrolling the page are the activities that causes an event to occur.

Types of Event

The events can be broadly classified into two categories:

- Foreground Events – These events require direct interaction of the user. They are generated as consequences of a person interacting with the graphical components in the Graphical User Interface.

  For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page, etc.

- Background Events – These events require the interaction of the end user. Operating system interrupts, hardware or software failure, timer expiration, and operation completion are some examples of background events

What is Event Handling?

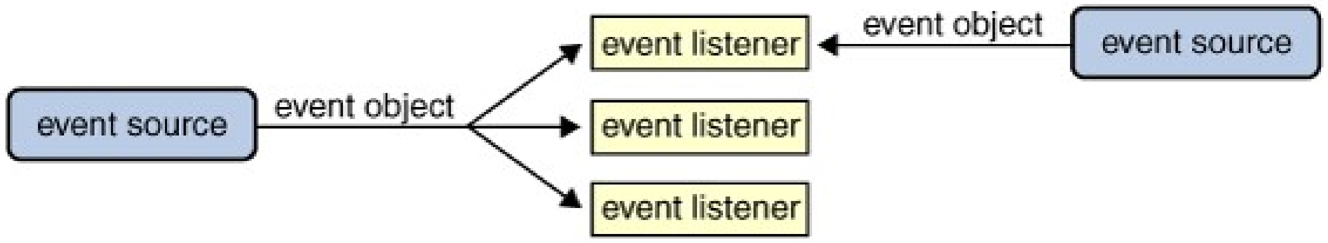
Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has a code which is known as an event handler, that is executed when an event occurs.

Java uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.

The Delegation Event Model has the following key participants.

- Source – The source is an object on which the event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide us with classes for the source object.
- Listener – It is also known as event handler. The listener is responsible for generating a response to an event. From the point of view of Java implementation, the listener is also an object. The listener waits till it receives an event. Once the event is received, the listener processes the event and then returns.

What is Event Handling?



Components in Event Handling

- Events
  - The events are defined as an object that describes a change in the state of a source object.
  - The Java defines a number of such Event Classes inside java.awt.event package
  - Some of the events are ActionEvent, MouseEvent, KeyEvent, FocusEvent, ItemEvent and etc.
- Event Sources
  - A source is an object that generates an event.
  - An event generation occurs when an internal state of that object changes in some way.
  - A source must register listeners in order for the listeners to receive the notifications about a specific type of event.
  - Some of the event sources are Button, CheckBox, List, Choice, Window and etc.
- Event Listeners
  - A listener is an object that is notified when an event occurs.
  - A Listener has two major requirements, it should be registered to one more source object to receiving event notification and it must implement methods to receive and process those notifications.
  - Java has defined a set of interfaces for receiving and processing the events under the java.awt.event package.
  - Some of the listeners are ActionListener, MouseListener, ItemListener, KeyListener, WindowListener and etc.

Important Event Classes and Interface

ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exits a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener

Important Event Classes and Interface

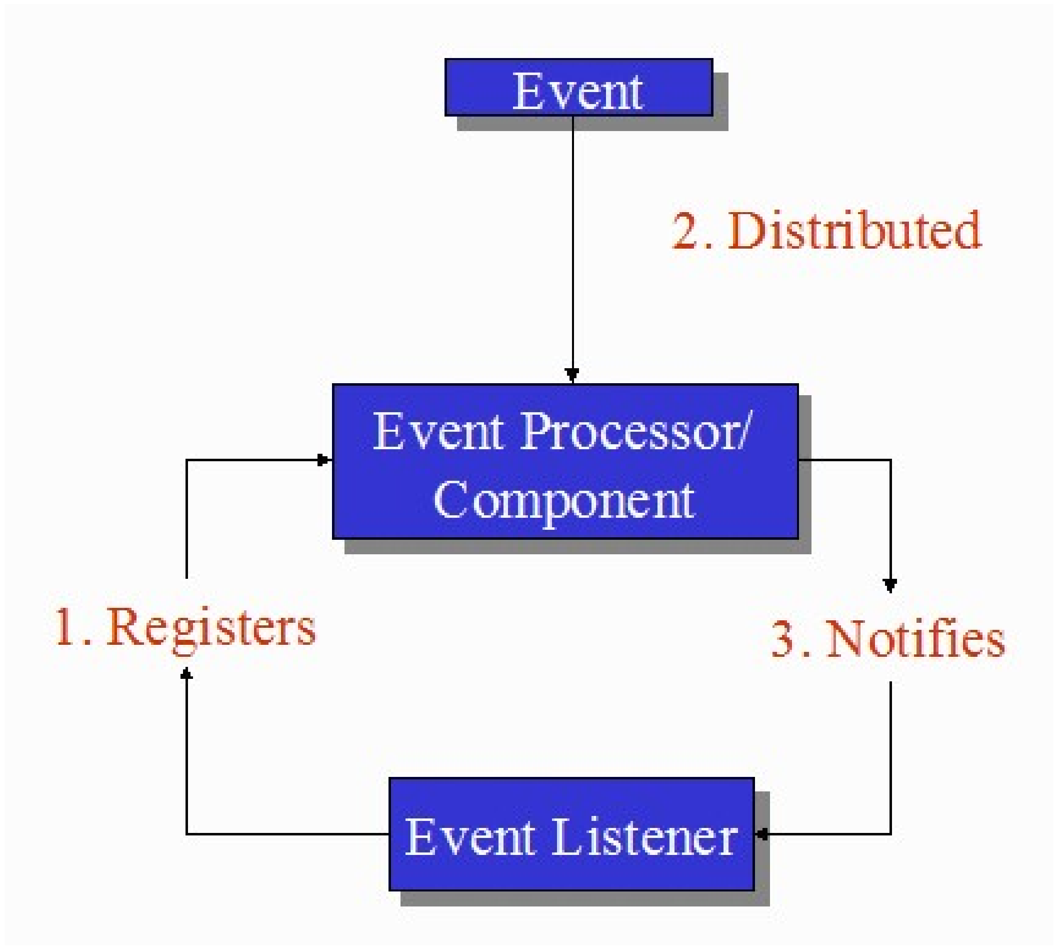
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Registration Methods

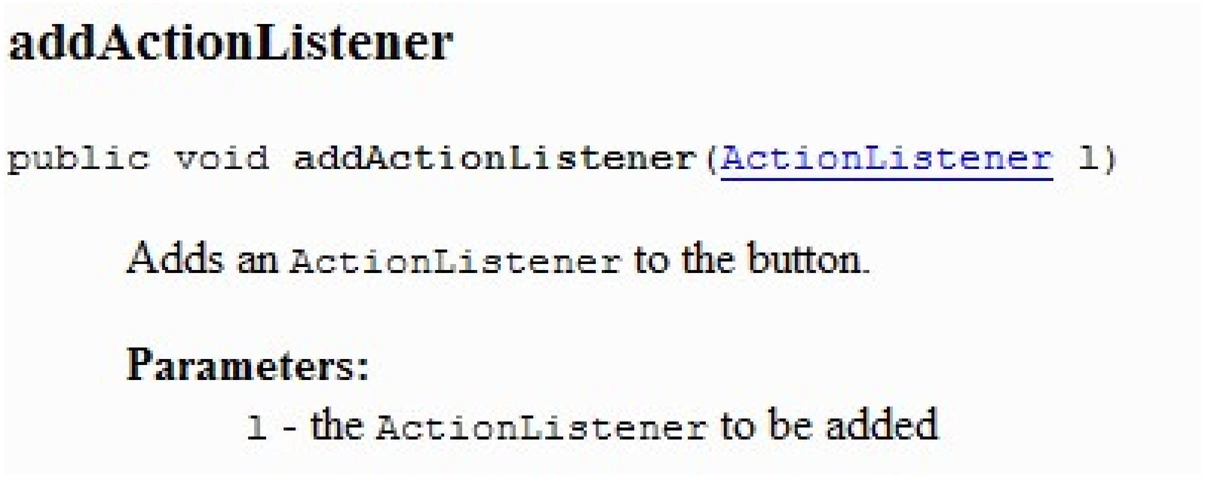
Register the component with the Listener

- Button
  - public void addActionListener(ActionListener a){}
- MenuItem
  - public void addActionListener(ActionListener a){}
- TextField
  - public void addActionListener(ActionListener a){}
  - public void addTextListener(TextListener a){}
- TextArea
  - public void addTextListener(TextListener a){}
- Checkbox
  - public void addItemListener(ItemListener a){}
- Choice
  - public void addItemListener(ItemListener a){}
- List
  - public void addActionListener(ActionListener a){}
  - public void addItemListener(ItemListener a){}

The Java Event Listener Pattern



Let's walk through the process of deciding to use a JButton in a component.



addActionListener

```
public void addActionListener(ActionListener l)
```

Adds an `ActionListener` to the button.

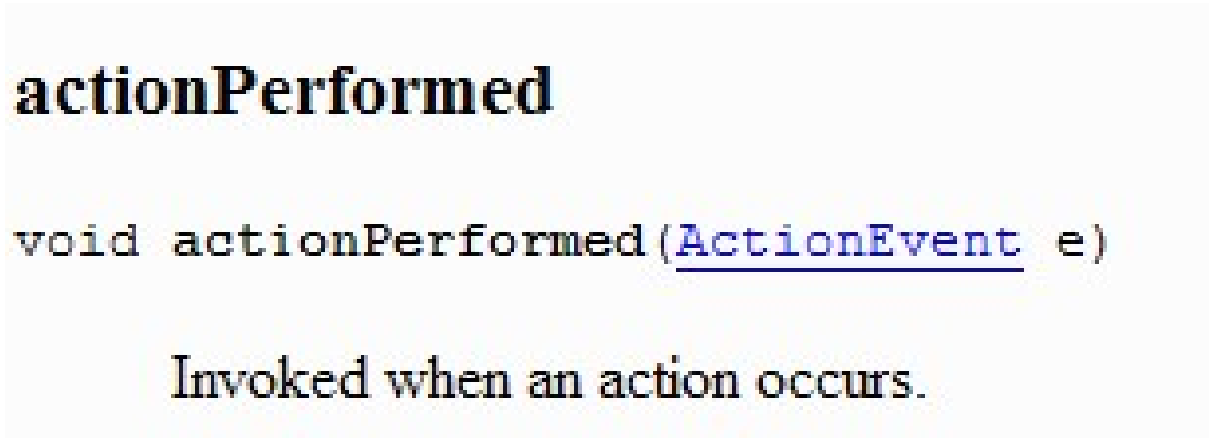
Parameters:

`l` - the `ActionListener` to be added

Step 1: See what listeners the JButton supports.

Step 2: create an object that implements the listener interface

For our JButton example, this is the ActionListener interface. If we look at the ActionListener interface we see a lone method that needs to be implemented.



actionPerformed

```
void actionPerformed(ActionEvent e)
```

Invoked when an action occurs.

This is the method that is called whenever the event that you are interested in happens. In JButton's case, whenever it is pushed, JButton will call this method in whatever class that registered for listening to this event.

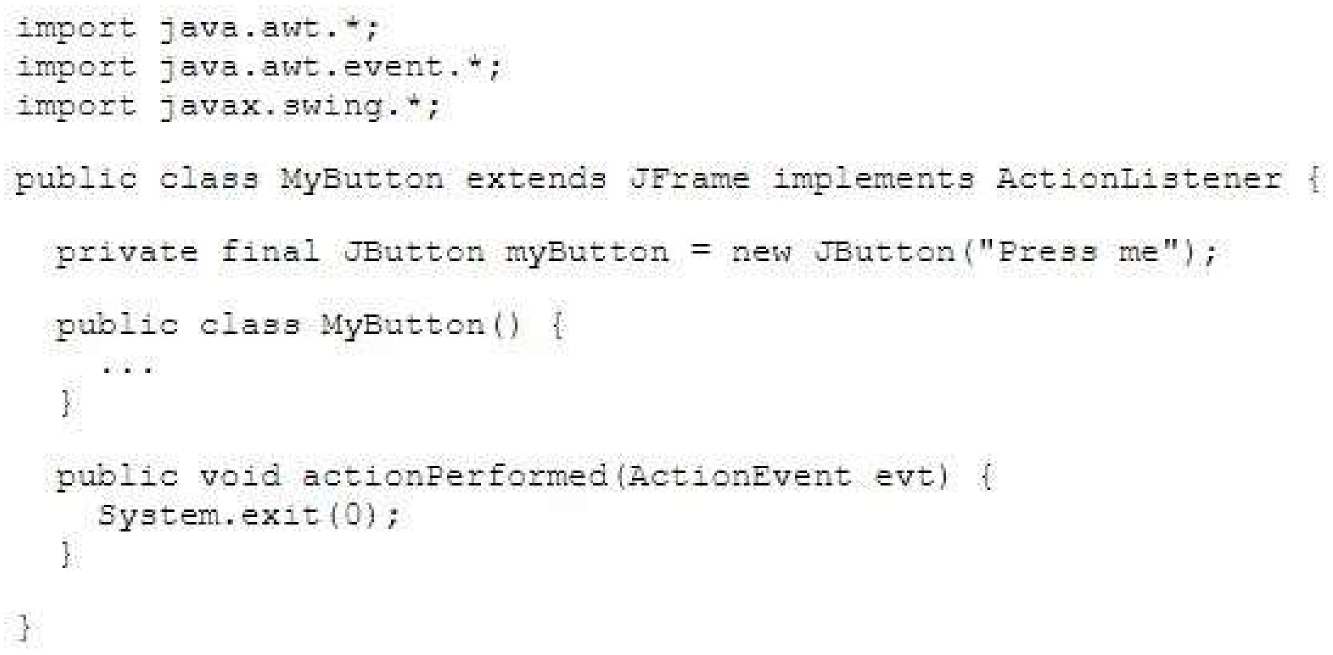
Step 2: create an object that implements the listener interface

There are many schemes to implement listeners. Some of the most common ones are

1. The component/container that holds the object you want to watch implements the interface
2. You have an inner class in the component/container that holds the object, and that inner class implements the interface
3. You have a separate class that implements the interface
4. You use an anonymous class to implement the interface

Step 2: create an object that implements the listener interface

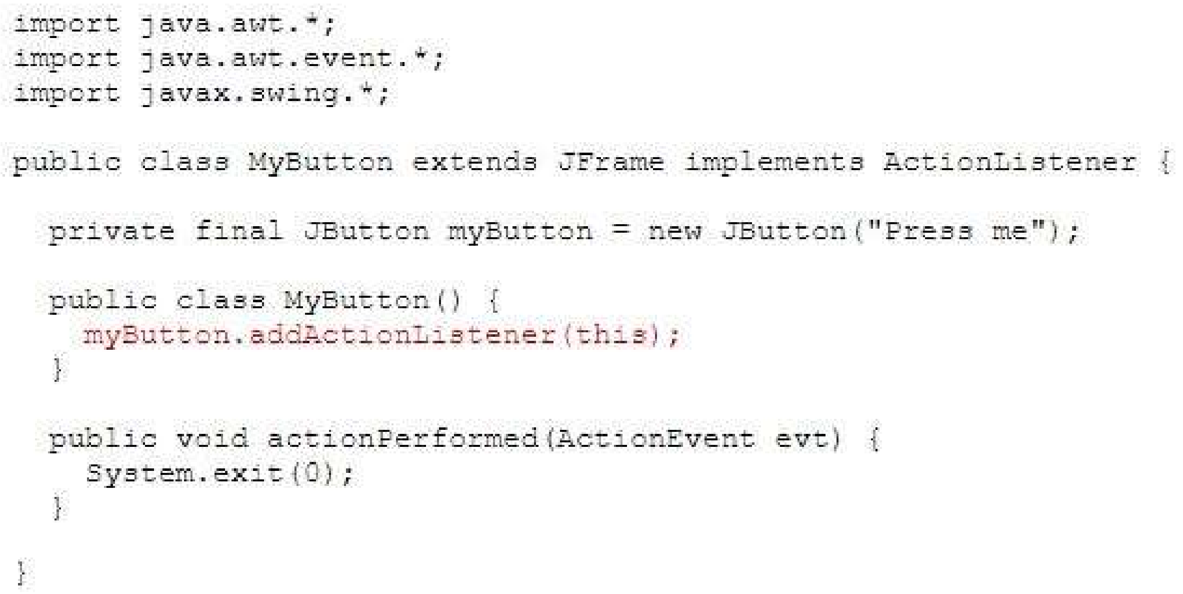
The class MyButton is a JFrame that implements the ActionListener interface. That means I have to have a method called public void actionPerformed(ActionEvent evt) in my MyButton class.



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

public class MyButton extends JFrame implements ActionListener {

  private final JButton myButton = new JButton("Press me");

  public class MyButton() {
    ...
  }

  public void actionPerformed(ActionEvent evt) {
    System.exit(0);
  }

}
```

Step 3: Register the object implementing the Listener interface with the Event Processing object

In this case, the Event Processing object is the JButton. It receives the Button Pushed event and notifies any listeners that an ActionEvent occurred. We register the Listener by adding it to the Event Processing object using the addActionListener() method



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

public class MyButton extends JFrame implements ActionListener {

  private final JButton myButton = new JButton("Press me");

  public class MyButton() {
    myButton.addActionListener(this);
  }

  public void actionPerformed(ActionEvent evt) {
    System.exit(0);
  }

}
```

Let's see some of the listeners

ActionListener

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

How to write ActionListener

1. Implement the ActionListener interface in the class:

   ```
   public class ActionListenerExample Implements ActionListener
   ```

2. Register the component with the Listener:

   ```
   component.addActionListener(instanceOfListenerclass);
   ```

3. Override the actionPerformed() method:

   ```
   public void actionPerformed(ActionEvent e){
           //Write the code here
   }
   ```

MouseListener

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

Methods of MouseListener interface

- public abstract void mouseClicked(MouseEvent e);
- public abstract void mouseEntered(MouseEvent e);
- public abstract void mouseExited(MouseEvent e);
- public abstract void mousePressed(MouseEvent e);
- public abstract void mouseReleased(MouseEvent e);

MouseMotionListener

The Java MouseMotionListener is notified whenever you move or drag mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java.awt.event package. It has two methods.

Methods of MouseMotionListener

- public abstract void mouseDragged(MouseEvent e);
- public abstract void mouseMoved(MouseEvent e);

ItemListener

The Java ItemListener is notified whenever you click on the checkbox. It is notified against ItemEvent. The ItemListener interface is found in java.awt.event package. It has only one method: itemStateChanged().

itemStateChanged() method

The itemStateChanged() method is invoked automatically whenever you click or unclick on the registered checkbox component.

```
public abstract void itemStateChanged(ItemEvent e);
```

KeyListener

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package. It has three methods.

Methods of KeyListener

- public abstract void keyPressed(KeyEvent e);
- public abstract void keyReleased(KeyEvent e);
- public abstract void keyTyped(KeyEvent e);

WindowListener

The Java WindowListener is notified whenever you change the state of window. It is notified against WindowEvent. The WindowListener interface is found in java.awt.event package. It has three methods.

Methods of WindowListener

- public abstract void windowActivated(WindowEvent e);
- public abstract void windowClosed(WindowEvent e);
- public abstract void windowClosing(WindowEvent e);
- public abstract void windowDeactivated(WindowEvent e);
- public abstract void windowDeiconified(WindowEvent e);
- public abstract void windowIconified(WindowEvent e);
- public abstract void windowOpened(WindowEvent e);

Next – A Swing application (*in site*)