Event handling

An event happens when something changes within a graphical user interface.
We can say that events are objects in Java. It comes under some classes stored in java.util.EvenObject.
For example, events occur when a user clicks on a button, clicks on a combo box, or types characters into a text field, etcetera, such as in the following:
For a button, the event that is fired is the ActionListener.
For a text field, it's the KeyEvent.
The following figure clarifies events. When we click on the "click me" button an event is generated; that change event generates another frame that shows our message that is passed to the program.

Listener

This class listens for the events in the application. It controls the application without affecting its internal mechanism.
The Listener interfaces check the continuity of the work, the dispatching of a class must be able to rely on each of its listeners to contain the method that is executed when the event occurs. It can be easily done in Java by the use of an Interface class. The important point is that a class, which is going to be a listener, must implement that interface. They are:
ServletContextListener and
HttpSessionListener.
When a listener is created, by the property of the interface "all the methods of that interface must be implemented". Some listeners, like the ActionListener, have only one method.
Event Handling
The Abstract Window Toolkit (AWT) uses event driven programming to do processing of user actions, one that underlies all modern window systems programming. Within the AWT, all user actions belong to an abstract set of things called events. An event describes, in sufficient detail, a particular user action. Rather than the program actively collecting user-generated

events, the Java run time notifies the program when an interesting event occurs. Programs that handle user interaction in this fashion are said to be event driven.
Event Handling
Event Handling provides four types of classes; they are:
Event Adapters
Event classes
Event Sources
Event Listeners
1. Event Adapters

In a program, when a listener has many abstract methods to override, it becomes complex for the programmer to override all of them.
For example, for closing a frame, we must override seven abstract methods of WindowListener, but we need only one method of them.
For reducing complexity, Java provides a class known as "adapters" or adapter class. Adapters are abstract classes, that are already being overriden.
2. Event classes

Every event source generates an event and is named by a Java class. An event is generated when something changes within a graphical user interface.
For example the event generated by a:
Button is known as an ActionEvent
Checkbox is known as an ItemEvent
All the events are listed in the java.awt.event package.
3. Event Sources

Event Sources are responsible for generating events and are called components.
The source for an event can be a button, TextField or a Frame etcetera.
Event Listeners

Events are handled by a special group of interfaces known as "listeners".

Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener

WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

How to perform event handling
The following is required to perform event handling:
Implement the Listener interface and override its methods
Register the component with the listener
For adding various components we use publics methods, for example:
Button
   void addActionListener(ActionListener a)
List
   void addActionListener(ActionListener a)
   void addItemListener(ItemListener a)
Choice
   void addItemListener(ItemListener x)
MenuItem
   void addActionListener(ActionListener x)
TextField
   void addActiontListener(ActionListener x)
   void addTextListener(TextListener x)
TextArea
   void addTextListener(TextListener x)
Checkbox
   void addItemListener(ItemListener x)
Java Examples on Delegation Event Model
MouseEvents class extends Applet and implements both the MouseListener and MouseMotionListener interfaces. These two interfaces contain methods that receive and process the various types of mouse events. KeyListener interface is found in java.awt.event package. KeyListener is notified whenever you change the state of key. This section contains Java programs on handling mouseevent and keyboardevent.

Keyboard event handling program

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

```
public class Test extends Applet implements KeyListener
{
 String msg="";
 public void init()
 {
  addKeyListener(this);
 }
 public void keyPressed(KeyEvent k)
 {
  showStatus("KeyPressed");
 }
 public void keyReleased(KeyEvent k)
 {
  showStatus("KeyRealesed");
 }
 public void keyTyped(KeyEvent k)
 {
  msg = msg+k.getKeyChar();
  repaint();
 }
 public void paint(Graphics g)
 {
  g.drawString(msg, 20, 40);
 }
}
```

Mouse event handling program

```
import java.awt.*;
import java.awt.event.*;
public class MouseApplet extends Applet implements MouseListener
{
        String msg="Initial Message";
        public void init()
        {
                addMouseListener(this);
        }
        public void mouseClicked(MouseEvent me)
        {
                msg = "Mouse Clicked";
                repaint();
        }
```

```
    public void mousePressed(MouseEvent me)
    {
        msg = "Mouse Pressed";
        repaint();
    }
    public void mouseReleased(MouseEvent me)
    {
        msg = "Mouse Released";
        repaint();
    }
    public void mouseEntered(MouseEvent me)
    {
        msg = "Mouse Entered";
        repaint();
    }
    public void mouseExited(MouseEvent me)
    {
        msg = "Mouse Exited";
        repaint();
    }
    public void paint(Graphics g)
    {
        g.drawString(msg,20,20);
    }
}
```

Reference

1. Mahesh Bhave and Sunil Patekar, "Programming with Java", First Edition, Pearson Education,2008, ISBN:9788131720806

2. Herbert Schildt, The Complete Reference C++, 4th Edition, Tata McGraw Hill, 2003.

3. Stanley B.Lippmann, Josee Lajore, C++ Primer, 4th Edition, Pearson Education, 2005.

4. Rajkumar Buyya,S Thamarasi selvi, xingchen chu, Object oriented Programming with java, Tata McGraw Hill education private limited.

5. Richard A Johnson, Introduction to Java Programming and OOAD, CENGAGE Learning.

6. E Balagurusamy, Programming with Java A primer, Tata McGraw Hill companies.

7. Sourav Sahay, Object Oriented Programming with C++ , 2nd Ed, Oxford University Press,2006 (Chapters 1, 2, 4)

8. Herbert Schildt, Java The Complete Reference, 7th Edition, Tata McGraw Hill, 2007. (Chapters 1, 2, 3, 4, 5, 6, 8, 9,10, 11, 21, 22, 29, 30)

9.www.c-sharpcorner.com

10.www.studytonight.com

11.www.javatpoint.com