Event Handling in Java

What is Event Handling?

Event Handling is a mechanism that controls the events and decides what should happen if an event occurs (like a button click, key press, mouse move, etc.).

Event Delegation Model in AWT

Java uses the **Event Delegation Model** to handle events in AWT. It consists of:

Component	Role
Event Source	The GUI component that generates events (e.g., Button, TextField).
Event Object	Encapsulates the event (e.g., `ActionEvent`, `MouseEvent`).
Event Listener	Interface that receives and handles the event (e.g., `ActionListener`).

Steps to Handle Events in AWT

1. **Implement the Listener Interface** (e.g., `ActionListener`)
2. **Register the Listener** with a component using `addXXXListener()` method.
3. **Override the required method** (e.g., `actionPerformed()`)

Common Event Classes and Listener Interfaces

Event Class	Listener Interface	Used For
`ActionEvent`	`ActionListener`	Button clicks, Menu items
`ItemEvent`	`ItemListener`	Checkbox/Choice selection
`TextEvent`	`TextListener`	Text changes in TextComponent
`KeyEvent`	`KeyListener`	Keyboard actions
`MouseEvent`	`MouseListener`, `MouseMotionListener`	Mouse clicks, moves
`WindowEvent`	`WindowListener`	Window open/close
`FocusEvent`	`FocusListener`	Component focus gain/loss

`AdjustmentEvent`	`AdjustmentListener`	Scrollbar adjustment

Example 1: Button Click using `ActionListener`

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

public class ButtonClickExample extends Frame implements ActionListener {
    Button b;

    ButtonClickExample() {
        b = new Button("Click Me");
        b.setBounds(100, 100, 80, 30);
        b.addActionListener(this); // Register the listener
        add(b);

        setSize(300, 300);
        setLayout(null);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        b.setLabel("Clicked");
    }

    public static void main(String[] args) {
        new ButtonClickExample();
    }
}
```

Example 2: Handling `TextEvent` from a TextField

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

public class TextEventExample extends Frame implements TextListener {
    TextField tf;

    TextEventExample() {
        tf = new TextField();
        tf.setBounds(100, 100, 150, 30);
        tf.addTextListener(this);

        add(tf);
        setSize(400, 400);
        setLayout(null);
        setVisible(true);
    }

    public void textValueChanged(TextEvent e) {
        System.out.println("Text Changed: " + tf.getText());
    }

    public static void main(String[] args) {
        new TextEventExample();
    }
}
```

Example 3: Handling Mouse Events

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

```
public class MouseEventExample extends Frame implements MouseListener {
    Label label;

    MouseEventExample() {
        label = new Label();
        label.setBounds(20, 50, 200, 30);
        addMouseListener(this);
        add(label);

        setSize(300, 300);
        setLayout(null);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        label.setText("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")");
    }
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}

    public static void main(String[] args) {
        new MouseEventExample();
    }
}
```

Window Closing Example (`WindowListener`)

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

```
public class WindowEventExample extends Frame {
    WindowEventExample() {
        setSize(300, 300);
        setLayout(null);
        setVisible(true);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.out.println("Window Closing...");
                dispose();
            }
        });
    }

    public static void main(String[] args) {
        new WindowEventExample();
    }
}
```

Using Adapter Classes (for Convenience)

If you don't want to implement all methods of a listener, use Adapter Classes like:

- `MouseAdapter`
- `KeyAdapter`
- `WindowAdapter`
- `FocusAdapter`

You can override only the methods you need.

Summary Table

Listener Interface	Event Class	Method to Override

`ActionListener`	`ActionEvent`	`actionPerformed(ActionEvent)`
`ItemListener`	`ItemEvent`	`itemStateChanged(ItemEvent)`
`TextListener`	`TextEvent`	`textValueChanged(TextEvent)`
`MouseListener`	`MouseEvent`	`mouseClicked(Mouse Event)`, `mousePressed(Mouse Event)`, `mouseReleased(Mouse Event)`, `mouseEntered(Mouse Event)`, `mouseExited(Mouse Event)`
`MouseMotionListener`	`MouseEvent`	`mouseDragged(Mouse Event)`, `mouseMoved(Mouse Event)`
`KeyListener`	`KeyEvent`	`keyPressed(KeyEvent e)`, `keyReleased(KeyEvent e)`, `keyTyped(KeyEvent e)`
`WindowListener`	`WindowEvent`	`windowClosing(WindowEvent e)`, `windowClosed(WindowEvent e)`, `windowIconifiedWindowEvent e)`, `windowDeiconified(WindowEvent e)`, `windowOpened(WindowEvent e)`, `windowActivated(WindowEvent e)`, `windowDeactivated(WindowEvent e)`,
`FocusListener`	`FocusEvent`	`focusGained(FocusEvent e)`, `focusLost(FocusEvent e)`

🔶 `MouseListener` Interface (Java AWT)

The `MouseListener` interface is part of `java.awt.event` package and is used to receive **mouse events** (like click, press, release, enter, and exit) on a component.

✅ Example Using All `MouseListener` Methods

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

public class MouseListenerExample extends Frame implements MouseListene
```

```
r {

    Label label;

    MouseListenerExample() {
        label = new Label();
        label.setBounds(20, 50, 250, 30);
        add(label);

        addMouseListener(this);  // Register MouseListener

        setSize(400, 300);
        setLayout(null);
        setVisible(true);
    }

    public void mouseClicked(MouseEvent e) {
        label.setText("Mouse Clicked at (" + e.getX() + ", " + e.getY() + ")");
    }

    public void mousePressed(MouseEvent e) {
        label.setText("Mouse Pressed");
    }

    public void mouseReleased(MouseEvent e) {
        label.setText("Mouse Released");
    }

    public void mouseEntered(MouseEvent e) {
        label.setText("Mouse Entered the frame");
    }

    public void mouseExited(MouseEvent e) {
        label.setText("Mouse Exited the frame");
    }
```

```
    public static void main(String[] args) {
        new MouseListenerExample();
    }
}
```
