Lab 9: Event Handling in Java

Instructor: Er. Bhawana Poudel

- Integral for GUI based programs
- Events are supported by packages like java.awt
- Response is generated when the user interacts with a GUI-based elements.

Event:

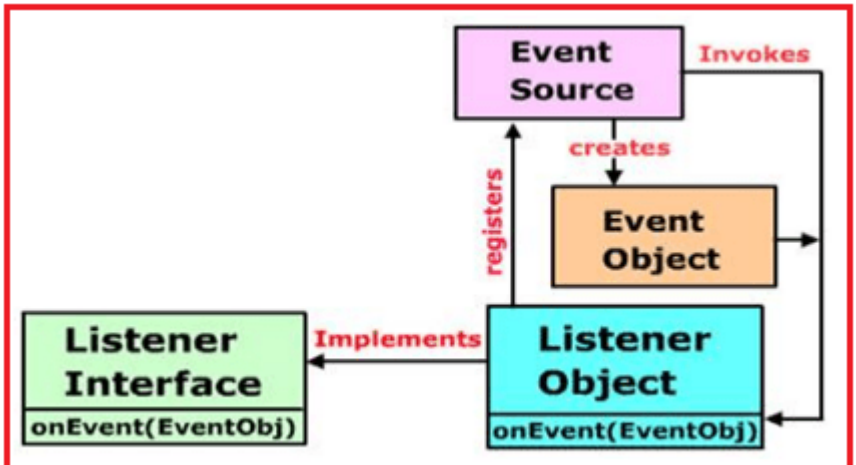
- change in the state of object or source

Event Handling:

- Controls the event and decides what should happen if an event occurs
- Delegation Event Model

Delegation Event Model

- Defines standard way for getting and processing events
- Mechanism:
  - Event Generation:
    - a source generates an event and sends it to listener(s)
    - E.g. Button Click event, etc
  - Event Listen & Handle:
    - Listeners waits until some event occurs, once an event is received, the listener processes the event and then returns.
    - Implement the interface in the listener so that it will receive the type of event desired .E.g. ActionListener is implemented for handling Button Click event.



Advantages of Delegation Event Model

- The processing of events is separated from the interface logic which generates those events
- An interface element is in a position to “delegate” the processing of an occasion to a separate piece of code.
- In the delegation event model, listeners must register with a source so as to receive an occasional notification:
    - Hence, notifications are sent only to listeners that want to receive them.
    - This is a more efficient way to handle events.

	Events	**Source Object**	**Listener Interface**	**Methods**
1	ActionEvent	Button, List, MenuItem, TextField	ActionListener	ActionPerformed( )
2	AdjustmentEvent	Component	ComponentListener	AdjustmentValueChanged( )
3	FocusEvent	Component	FocusListener	focusGained( ) focusLost( )
4	TextEvent	Text Component	TextListener	TextChanged( )
5	ItemEvent	Checkbox,choice	ItemListener	ItemStateChanged( )
6	MouseEvent	Mouse Movement	MouseListener	MousePressed( ) mouseClicked( ) mouseEntered( ) mouseExited( ) mouseReleased( )
7	WindowEvent	Window	WindowListener	windowActivated( ) windowDeactivated( )

				windowOpened( ) windowClosed( ) windowClosing( )
8	KeyEvent	TextComponent	KeyListener	keyTyped( ) keyReleased() keyPressed( )

1.1 Example showing Steps to Handle Event- ActionListener

Let us discuss about click event handling in a button

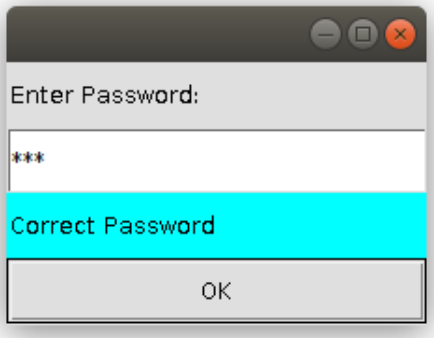
1. **Event Generation:** Whenever the user clicks the button an event is generated.

2. **Object Creation:** Object of the concerned event class will be automatically created and information about the source and the event gets populated within the same object.

3. **Listener Invocation:** Then the event object is forwarded to the method of the registered listener class.

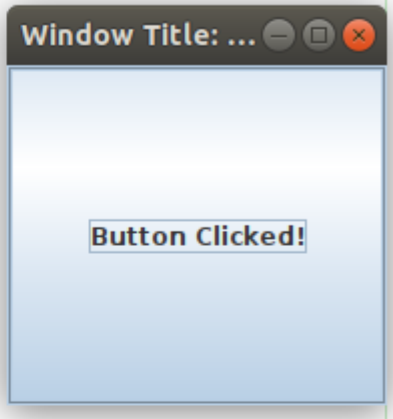
4. **Process Event:** Now the method will get executed and returned.

```
1
2  import java.awt.*;
3  import java.awt.event.*;
4  public class CheckPassword extends Frame implements ActionListener{
5      Button b; Label l1,l2;
6      TextField t;
7      GridLayout glay;
8      CheckPassword(){
9
10         b = new Button("OK");
11         l1=new Label("Enter Password:");
12         l2= new Label();
13         t= new TextField(10);
14         t.setEchoChar('*');
15         glay=new GridLayout(4,1);
16         setSize(200,150);
17         setLayout(glay);
18         add(l1);     add(t);
19         add(l2);     add(b);
20         b.addActionListener(this);
21     }
22
23     public void actionPerformed(ActionEvent e){
24         String s= e.getActionCommand();
25         String psw;
26         if(s.equals("OK")){
27             psw=t.getText();
28             if(psw.equals("BIT")){
29                 l2.setText("Correct Password");
30                 l2.setBackground (Color.cyan);
31             }
32             else{
33                 l2.setText("Inorrect Password");
34                 l2.setBackground (Color.red);
35             }
36         }
37     }
38     public static void main(String args[]){
39         CheckPassword cp= new CheckPassword();
40         cp.setVisible(true);
41     }
42 }
43
```



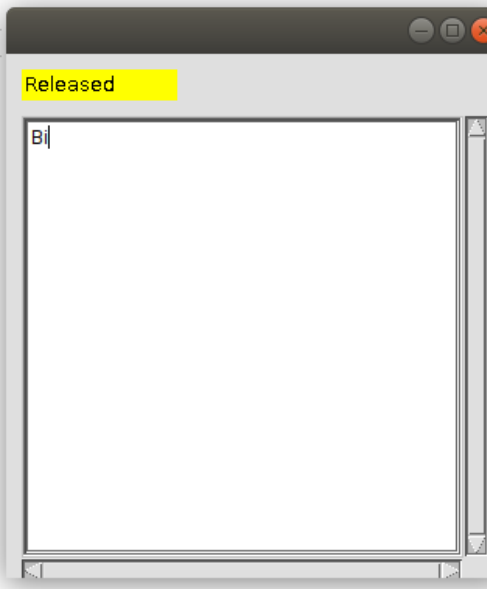
1.2 MyEvent.java : same example BUT a new outer class implements the ActionListener interface

```
1  //Write a program uisng the swing to handle the mouse click on Frame.
2  //Add a button in the Frame. Implement ActionListener so that when you
3  //click on the button change the display text of the button.
4
5  import javax.swing.*;
6  import java.awt.event.*;
7
8  public class MyEvent extends JFrame
9  {
10     JButton b1;
11     // Main Method
12      public static void main (String arg[])
13      {
14          MyEvent event = new MyEvent();
15       }
16
17   //Constructor for the event derived class
18    public MyEvent()
19    {
20          super("Window Title: Event Handling");
21           b1 = new JButton("Click Me");
22           //place the button object on the window
23           getContentPane().add(b1);
24
25            //Register the listener for the button
26           ButtonListener listen = new ButtonListener();
27           b1.addActionListener(listen);
28           //display the window in a specific size
29           setVisible(true);
30           setSize(200,200);
31      }
32 //The Listener Class
33   class ButtonListener implements ActionListener
34    {
35        //Definition for ActionPerformed() method
36         public void actionPerformed(ActionEvent evt)
37         {
38              JButton source = (JButton)evt.getSource();
39               source.setText("Button Clicked!");
40          }
41     }
42
43 }
44
```



2.1 Example showing KeyListener Example

```java
import java.awt.*;
import java.awt.event.*;
// class which inherits Frame class and implements KeyListener interface
public class KeyListenerExample extends Frame implements KeyListener {

    Label l;
    TextArea area;

    KeyListenerExample() {
        l = new Label();
        // setting the location of the label in frame
        l.setBounds (10, 40, 100, 20);
        area = new TextArea();
        area.setBounds (10, 70, 300, 300);

        // adding the label and text area to the frame
        add(l);
        add(area);

        // adding the KeyListener to the text area
        area.addKeyListener(this);

        // Define the size, layout and visibility of frame
        setSize (500, 400);
        setLayout (null);// Comment this line and default flowlayout will apply
        setVisible (true);
    }

    // overriding the keyPressed() method of KeyListener interface AND set the text of the label when key is pres
    public void keyPressed (KeyEvent e) {
        l.setText ("Pressed");
        l.setBackground (Color.cyan);
    }

    // overriding the keyReleased() method of KeyListener interface AND set the text of the label when key is rel
    public void keyReleased (KeyEvent e) {
        l.setText ("Released");
        l.setBackground (Color.yellow) ;
    }

    // overriding the keyTyped() method of KeyListener interface AND set the text of the label when a key is type
    public void keyTyped (KeyEvent e) {
        //l.setText ("Typed");
        //l.setBackground (Color.pink);
    }

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



2.2 KeyPress.java : same example BUT a new class implements the KeyListener

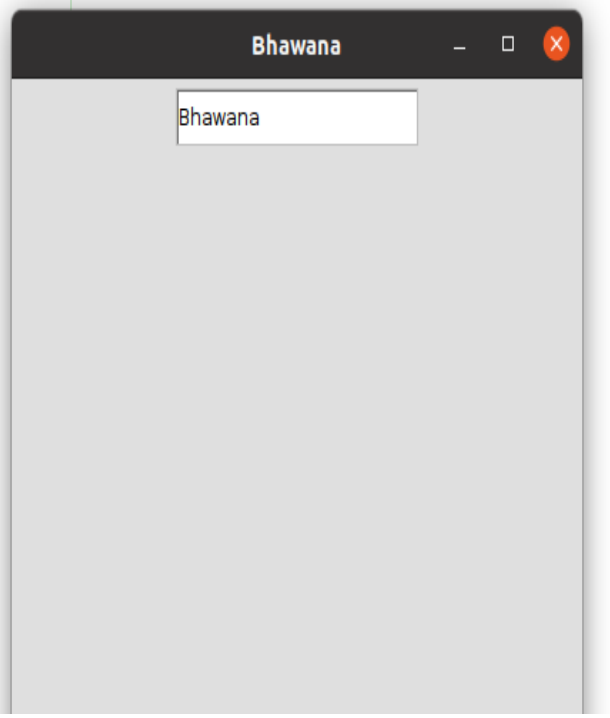
```java
//Write a program which receives the generated event when you press
// any key to the object and displays it.

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

public class KeyPress extends Frame{
  Label label;
  TextField txtField;
  public static void main(String[] args) {
    KeyPress k = new KeyPress();
  }

  public KeyPress(){
    super("Key Press Event Frame");
    Panel panel = new Panel(); //#1
    label = new Label();
    txtField = new TextField(20);
    txtField.addKeyListener(new MyKeyListener()); //
    add(label, BorderLayout.NORTH);
    panel.add(txtField, BorderLayout.CENTER); // #2
    add(panel, BorderLayout.CENTER);//#3
    //add(txtField, BorderLayout.CENTER);
    /* you can directly add the txtField without
       adding it to the panel(removing #1, #2 and #3)
    */
    //want ot close the window by clicking on the cross
    addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent we){
        System.exit(0);
      }
    });

    setSize(400,400);
    setVisible(true);
  }

  // class which inherits Frame class and implements KeyListener interface
  public class MyKeyListener extends KeyAdapter{
    public void keyPressed(KeyEvent ke){
      char i = ke.getKeyChar();
      String str = Character.toString(i);
      label.setText(str);
    }
  }
}
```

interface

Line 28: WindowEvent (implementing WindowListener) is also shown in this example 2.2

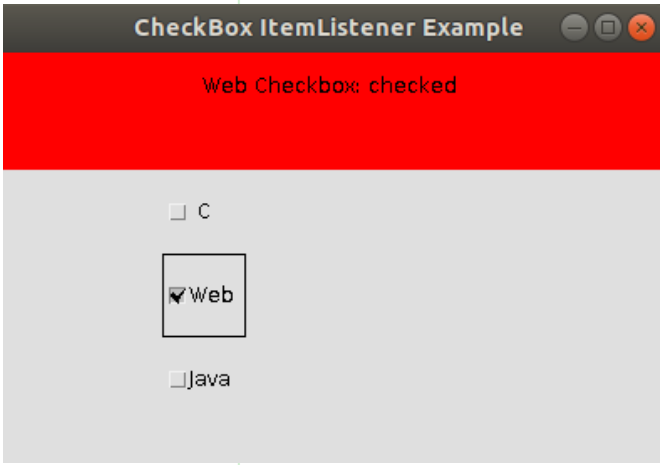
. Example showing TextListener Example

```
import java.awt.*;
import java.awt.event.*;
class TextListenerExample extends Frame implements TextListener
{
TextField txt;
public TextListenerExample()
{
setTitle("Example of Text Listener");
setLayout(new FlowLayout());
txt=new TextField(20);
add(txt);
txt.addTextListener(this);
setSize(400,400);
setVisible(true);
}
public void textValueChanged(TextEvent e)
{
setTitle(txt.getText());
}
public static void main(String[] args)
{
new TextListenerExample();
}
}
```



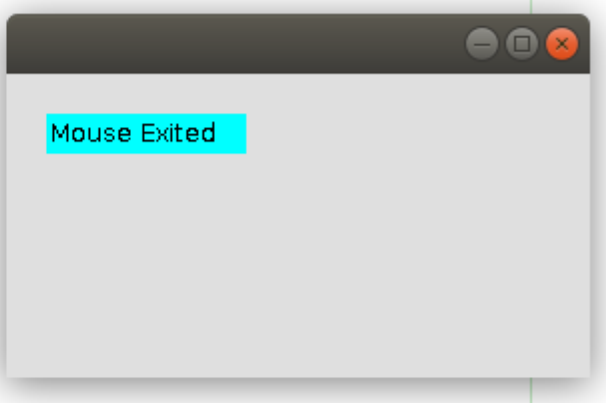
4. Example showing CheckboxItemListener Example

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

public class CheckboxItemListenerExample implements ItemListener{

    Checkbox checkBox1, checkBox2, checkBox3;
    Label label;
    CheckboxItemListenerExample(){
        Frame f= new Frame("CheckBox ItemListener Example");
        label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);

        checkBox1 = new Checkbox("C");
        checkBox1.setBounds(100,100, 50,50);
        checkBox2 = new Checkbox("Web");
        checkBox2.setBounds(100,150, 50,50);
        checkBox3 = new Checkbox("Java");
        checkBox3.setBounds(100,200, 50,50);
        f.add(checkBox1);       f.add(checkBox2);
        f.add(checkBox3);       f.add(label);

        checkBox1.addItemListener(this);
        checkBox2.addItemListener(this);
        checkBox3.addItemListener(this);

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

    public void itemStateChanged(ItemEvent e) {
        if(e.getSource()==checkBox1){
            label.setText("C Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
            label.setBackground (Color.cyan);
        }

        if(e.getSource()==checkBox2)  {
            label.setText("Web Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
            label.setBackground (Color.red);
        }
        if(e.getSource()==checkBox3)  {
            label.setText("Java Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
            label.setBackground (Color.yellow);
        }
    }

    public static void main(String args[]) {
        CheckboxItemListenerExample obj = new CheckboxItemListenerExample();
    }
}
```



5. **Example showing MouseEvent Example**

```java
import java.awt.*;
import java.awt.event.*;
public class MouseEventExample extends Frame implements MouseListener{

    Label label;

    MouseEventExample(){

        label=new Label();
        label.setBounds(20,50,100,20);
        add(label);

        addMouseListener(this);

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

    public void mouseClicked(MouseEvent e) {
        label.setText("Mouse Clicked");
    }
    public void mouseEntered(MouseEvent e) {
        label.setText("Mouse Entered");
    }
    public void mouseExited(MouseEvent e) {
        label.setText("Mouse Exited");
    }
    public void mousePressed(MouseEvent e) {
        label.setText("Mouse Pressed");
    }
    public void mouseReleased(MouseEvent e) {
        label.setText("Mouse Released");
         label.setBackground (Color.cyan);
    }

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

```



6. Example showing FocusListener Example

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

public class FocusListenerExample extends Frame implements FocusListener
    {
     Button b1,b2;
     Label l1,l2;

     public FocusListenerExample()
         {
         add(b1=new Button ("First"),"North");
         add(b2=new Button ("Second"),"South");
         add(l1=new Label ("See Focus Gained MSG"),"East");
         add(l2=new Label ("See Focus Lost MSG"),"West");
         b1.addFocusListener(this);
         b2.addFocusListener(this);
         setSize(400,300);
     }

     public void focusGained(FocusEvent fEvnt)
    {
         if(fEvnt.getSource()==b1)
         l1.setText(b1.getLabel()+" Start");
         if(fEvnt.getSource()==b2)
         l1.setText(b2.getLabel()+" Start");
         if(fEvnt.isTemporary())
         l1.setText("Temporary Focus");
     }

     public void focusLost(FocusEvent fEvnt)
    {
         if(fEvnt.getSource()==b1)
         l2.setText(b1.getLabel()+" Ended");
         if(fEvnt.getSource()==b2)
         l2.setText(b2.getLabel()+" Ended");
     }

     public static void main(String a[])
     {
         new FocusListenerExample().setVisible(true);
     }
}

```

7. Example showing AdjustmentListener Example

- Create two scrollbars and display the sum of their values in a TextField

```
1  import java.awt.*;
2  import java.awt.event.*;
3  public class ScrollbarDemo extends Frame implements AdjustmentListener{
4      Scrollbar sb1,sb2;
5      TextField tf;
6      Label l;
7      
8      public ScrollbarDemo(){
9          
10         sb1= new Scrollbar(Scrollbar.VERTICAL,0,0,1,500);
11         
12         sb2= new Scrollbar(Scrollbar.HORIZONTAL,0,0,1,500);
13         tf= new TextField(10);
14         l = new Label("Sum");
15         setLayout(new FlowLayout());
16         add(sb1);
17         add(sb2);
18         add(l);
19         add(tf);
20         
21         setSize(300,150);
22         sb1.addAdjustmentListener(this);
23         sb2.addAdjustmentListener(this);
24     }
25     
26     public void adjustmentValueChanged(AdjustmentEvent e){
27     int a= sb1.getValue();
28     int b= sb2.getValue();
29     int sum;
30     sum=a+b;
31     tf.setText(""+sum+"");
32     }
33     public static void main(String []args){
34     ScrollbarDemo sd= new ScrollbarDemo();
35     sd.setVisible(true);
36     }
37 }
38
```