

## Lab 8: Event Handling in Java

- 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:

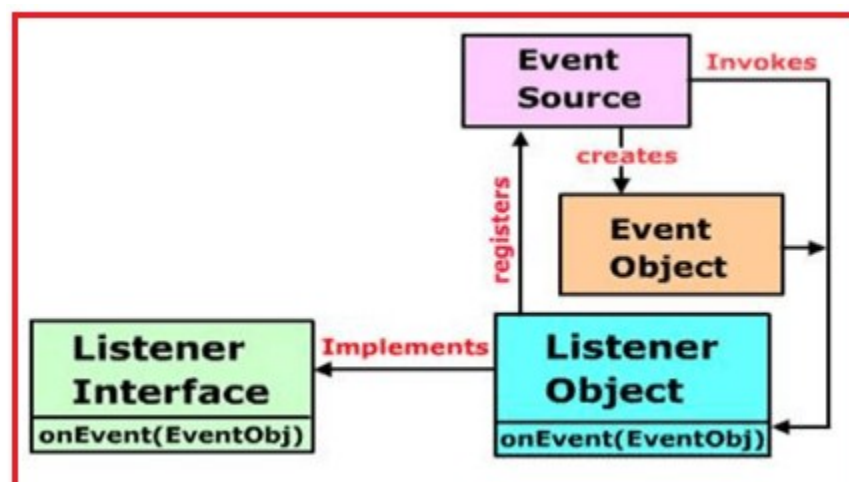
- change in the state of object or source

### 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.



<https://t.ly/CxINF>

### Advantages of Delegation Event Model

- Appliance logic( i.e. 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( )

## Lab 8: Event Handling in Java

				keyReleased() keyPressed( )
--	--	--	--	--------------------------------

## 1.1 Example showing Steps to Handle Event- ActionListener

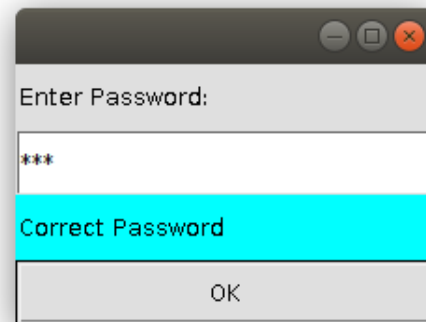
Let us discuss about click event handling in a button

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

```

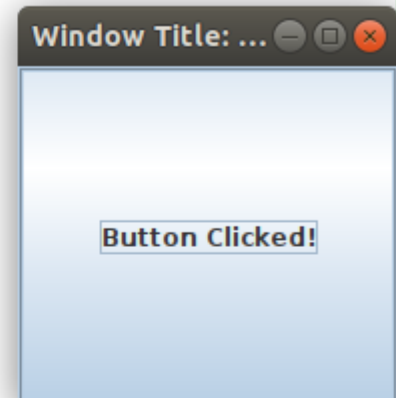


## 1.2 MyEvent.java : same example BUT a new outer class implements the ActionListener interface

```

1 //Write a program using 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
33     //The Listener Class
34     class ButtonListener implements ActionListener
35     {
36         //Definition for actionPerformed() method
37         public void actionPerformed(ActionEvent evt)
38         {
39             JButton source = (JButton)evt.getSource();
40             source.setText("Button Clicked!");
41         }
42     }
43 }
44

```



## 2.1 Example showing **KeyListener** Example

```

1  import java.awt.*;
2  import java.awt.event.*;
3  // class which inherits Frame class and implements KeyListener interface
4  public class KeyListenerExample extends Frame implements KeyListener {
5
6      Label l;
7      TextArea area;
8
9      KeyListenerExample() {
10         l = new Label();
11         // setting the location of the label in frame
12         l.setBounds(10, 40, 100, 20);
13         area = new TextArea();
14         area.setBounds(10, 70, 300, 300);
15
16         // adding the label and text area to the frame
17         add(l);
18         add(area);
19
20         // adding the KeyListener to the text area
21         area.addKeyListener(this);
22
23         // Define the size, layout and visibility of frame
24         setSize(500, 400);
25         setLayout(null); // Comment this line and default flowlayout will apply
26         setVisible(true);
27     }
28
29     // overriding the keyPressed() method of KeyListener interface AND set the text of the label when key is pressed
30     public void keyPressed (KeyEvent e) {
31         l.setText ("Pressed");
32         l.setBackground (Color.cyan);
33     }
34
35     // overriding the keyReleased() method of KeyListener interface AND set the text of the label when key is released
36     public void keyReleased (KeyEvent e) {
37         l.setText ("Released");
38         l.setBackground (Color.yellow);
39     }
40
41     // overriding the keyTyped() method of KeyListener interface AND set the text of the label when a key is typed
42     public void keyTyped (KeyEvent e) {
43         //l.setText ("Typed");
44         //l.setBackground (Color.pink);
45     }
46
47     public static void main(String[] args) {
48         new KeyListenerExample();
49     }
50 }
51

```

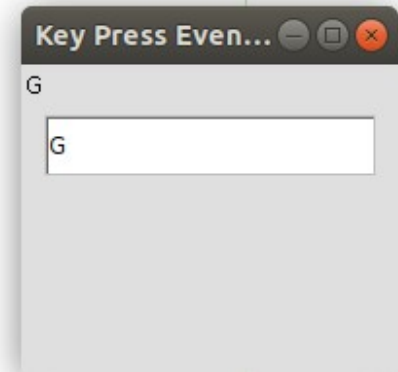


## 2.2 KeyPress.java : same example BUT a new class implements the KeyListener interface

```

1  //Write a program which receives the generated event when you press
2  // any key to the object and displays it.
3
4  import java.awt.*;
5  import java.awt.event.*;
6
7  public class KeyPress extends Frame{
8      Label label;
9      TextField txtField;
10     public static void main(String[] args) {
11         KeyPress k = new KeyPress();
12     }
13
14     public KeyPress(){
15         super("Key Press Event Frame");
16         Panel panel = new Panel(); // #1
17         label = new Label();
18         txtField = new TextField(20);
19         txtField.addKeyListener(new MyKeyListener()); //
20         add(label, BorderLayout.NORTH);
21         panel.add(txtField, BorderLayout.CENTER); // #2
22         add(panel, BorderLayout.CENTER); // #3
23         //add(txtField, BorderLayout.CENTER);
24         /* you can directly add the txtField without
25         adding it to the panel (removing #1, #2 and #3)
26         */
27         //want to close the window by clicking on the cross
28         addWindowListener(new WindowAdapter(){
29             public void windowClosing(WindowEvent we){
30                 System.exit(0);
31             }
32         });
33
34         setSize(400,400);
35         setVisible(true);
36     }
37
38     // class which inherits Frame class and implements KeyListener interface
39     public class MyKeyListener extends KeyAdapter{
40         public void keyPressed(KeyEvent ke){
41             char i = ke.getKeyChar();
42             String str = Character.toString(i);
43             label.setText(str);
44         }
45     }
46 }
47

```



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

### 3. Example showing **TextListener Example**

```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  class TextListenerExample extends Frame implements TextListener
5  {
6      TextField txtField;
7      public TextListenerExample()
8      {
9          setTitle("Example of Text Listener");
10         setLayout(new FlowLayout());
11         txtField=new TextField(20);
12
13         add(txtField);
14
15         txtField.addTextListener(this);
16         setSize(400,400);
17         setVisible(true);
18     }
19
20     public void textValueChanged(TextEvent e)
21     {
22         setTitle(txtField.getText());
23     }
24
25     public static void main(String args[])
26     {
27         new TextListenerExample();
28     }
29 }
30
```



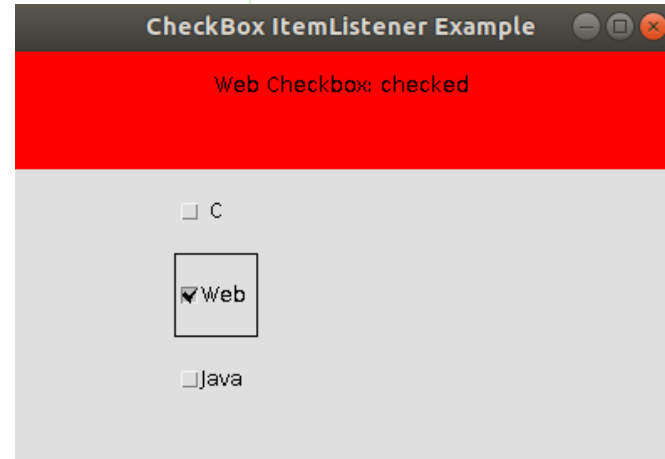


4. Example showing **CheckboxItemListener** Example

```

1  import java.awt.*;
2  import java.awt.event.*;
3
4  public class CheckboxItemListenerExample implements ItemListener{
5
6      ... Checkbox checkBox1, checkBox2, checkBox3;
7      ... Label label;
8
9      ... CheckboxItemListenerExample(){
10         ... Frame f= new Frame("CheckBox ItemListener Example");
11         ... label = new Label();
12         ... label.setAlignment(Label.CENTER);
13         ... label.setSize(400,100);
14
15         ... checkBox1 = new Checkbox("C");
16         ... checkBox1.setBounds(100,100, 50,50);
17         ... checkBox2 = new Checkbox("Web");
18         ... checkBox2.setBounds(100,150, 50,50);
19         ... checkBox3 = new Checkbox("Java");
20         ... checkBox3.setBounds(100,200, 50,50);
21         ... f.add(checkBox1); f.add(checkBox2);
22         ... f.add(checkBox3); f.add(label);
23
24         ... checkBox1.addItemListener(this);
25         ... checkBox2.addItemListener(this);
26         ... checkBox3.addItemListener(this);
27
28         ... f.setSize(400,300);
29         ... f.setLayout(null);
30         ... f.setVisible(true);
31     }
32
33     public void itemStateChanged(ItemEvent e) {
34         if(e.getSource()==checkBox1){
35             label.setText("C Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
36             label.setBackground(Color.cyan);
37         }
38         if(e.getSource()==checkBox2){
39             label.setText("Web Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
40             label.setBackground(Color.red);
41         }
42         if(e.getSource()==checkBox3){
43             label.setText("Java Checkbox: " + (e.getStateChange()==1?"checked":"unchecked"));
44             label.setBackground(Color.yellow);
45         }
46     }
47
48     public static void main(String args[]){
49         CheckboxItemListenerExample obj = new CheckboxItemListenerExample();
50     }
51 }

```



5. Example showing **MouseEvent** Example

```

1  import java.awt.*;
2  import java.awt.event.*;
3  public class MouseEventExample extends Frame implements MouseListener{
4      ...
5      Label label;
6      ...
7      MouseEventExample(){
8          ...
9          label=new Label();
10         label.setBounds(20,50,100,20);
11         add(label);
12         ...
13         addMouseListener(this);
14         ...
15         setSize(400,300);
16         setLayout(null);
17         setVisible(true);
18     }
19     ...
20     public void mouseClicked(MouseEvent e){
21         label.setText("Mouse Clicked");
22     }
23     public void mouseEntered(MouseEvent e){
24         label.setText("Mouse Entered");
25     }
26     public void mouseExited(MouseEvent e){
27         label.setText("Mouse Exited");
28     }
29     public void mousePressed(MouseEvent e){
30         label.setText("Mouse Pressed");
31     }
32     public void mouseReleased(MouseEvent e){
33         label.setText("Mouse Released");
34         label.setBackground(Color.cyan);
35     }
36     ...
37     public static void main(String[] args){
38         MouseEventExample obj = new MouseEventExample();
39     }
40 }
41

```

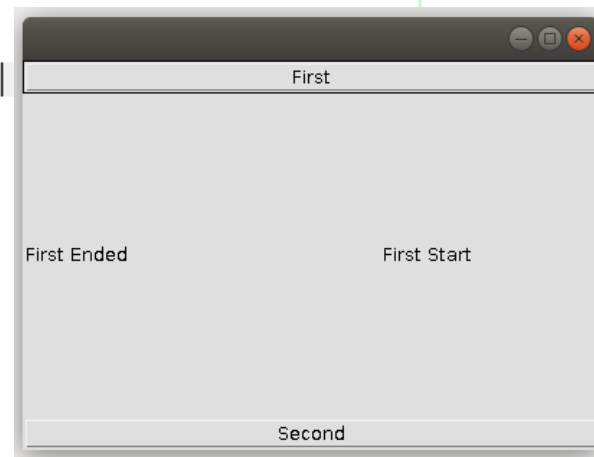


6. Example showing **FocusListener** Example

```

1  import java.awt.*;
2  import java.awt.event.*;
3
4  public class FocusListenerExample extends Frame implements FocusListener
5  {
6      Button b1,b2;
7      Label l1,l2;
8
9      public FocusListenerExample()
10     {
11         add(b1=new Button ("First"), "North");
12         add(b2=new Button ("Second"), "South");
13         add(l1=new Label ("See Focus Gained MSG"), "East");
14         add(l2=new Label ("See Focus Lost MSG"), "West");
15         b1.addFocusListener(this);
16         b2.addFocusListener(this);
17         setSize(400,300);
18     }
19
20     public void focusGained(FocusEvent fEvt)
21     {
22         if(fEvt.getSource()==b1)
23             l1.setText(b1.getLabel()+" Start");
24         if(fEvt.getSource()==b2)
25             l1.setText(b2.getLabel()+" Start");
26         if(fEvt.isTemporary())
27             l1.setText("Temporary Focus");
28     }
29
30     public void focusLost(FocusEvent fEvt)
31     {
32         if(fEvt.getSource()==b1)
33             l2.setText(b1.getLabel()+" Ended");
34         if(fEvt.getSource()==b2)
35             l2.setText(b2.getLabel()+" Ended");
36     }
37
38     public static void main(String a[])
39     {
40         new FocusListenerExample().setVisible(true);
41     }
42 }
43
44

```



## 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

```

