Module IV

Part IV

Java GUI Event Handling

- Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven.
- Event describes the change in state of any object.
- **For Example :** Pressing a button, Clicking or Dragging a mouse, etc.

Types of Event

- The events can be broadly classified into two categories:
- **Foreground Events**
  - Those events which require the direct interaction of user.
  - They are generated as consequences of a person interacting with the graphical components in 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**
  - Those events that require the interaction of end user are known as background events.
  - Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Components of Event Handling

- Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.
- This mechanism have the code which is known as event handler that is executed when an event occurs.
- Event handling has three main components,
  - **Events :** An event is a change in state of an object.
  - **Events Source :** Event source is an object that generates an event.
  - **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.

How Events are handled?

- A source generates an Event and send it to one or more listeners registered with the source.
- Once event is received by the listener, they process the event and then return.
- In Java events are supported by **java.awt.event** package

The java.awt.event Package

- The java.awt.event package defines classes and interfaces used for event handling in the AWT.
- **The members of this package fall into three categories:**

1. Events:

- The classes with names ending in "Event" represent specific types of events, generated by the AWT or by one of the AWT components.

2. Listeners:

- The interfaces in this package are all event listeners; their names end with "Listener". These interfaces define the methods that must be implemented by any object that wants to be notified when a particular event occurs.
- Note that there is a Listener interface for each Event class.

3. Adapters :

- Each of the classes with a name ending in "Adapter" provides a no-op implementation for an event listener interface that defines more than one method.
- When you are interested in only a single method of an event listener interface, it is easier to subclass an Adapter class than to implement all of the methods of the corresponding Listener interface.

Important Event Classes and Interfaces

Event Classes	**Description**	**Listener 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,presse d or released and also when it enters or exit 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
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

Java ActionListener Interface

- The Java ActionListener is notified whenever you click on the button or menu item.
- It is notified against ActionEvent.
- It has only one method: actionPerformed().
- The actionPerformed() method is invoked automatically whenever you click on the registered component.

How to write ActionListener

- If you implement the ActionListener class, you need to follow 3 steps:
- 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 }

EXAMPLE

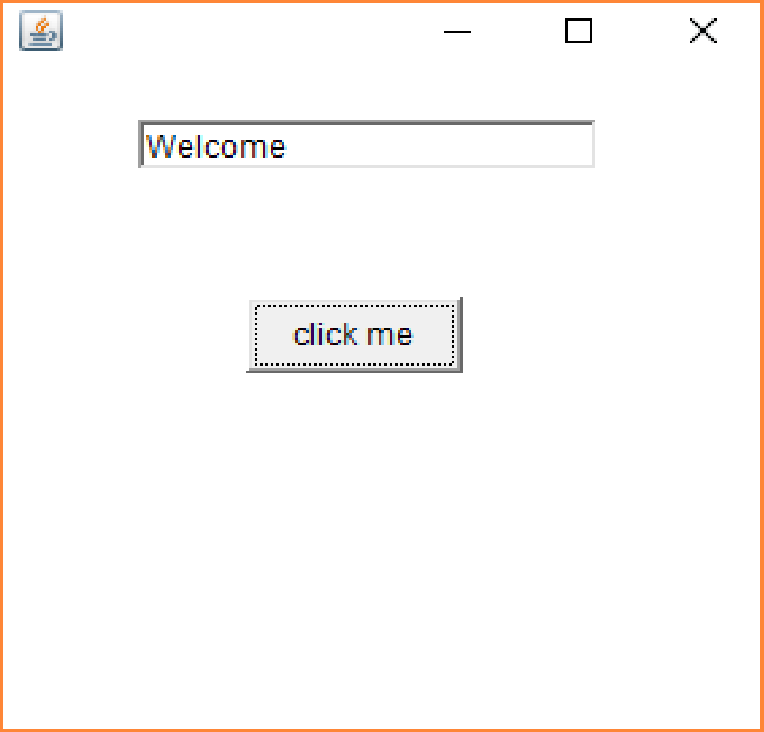
```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener{ //implementing ActionListener
TextField tf;
AEvent(){

//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
Button b=new Button("click me");
b.setBounds(100,120,80,30);

//register listener
b.addActionListener(this);//passing current instance

//add components and set size, layout and visibility
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){  //overriding the method actionPerformed
tf.setText("Welcome");
}
public static void main(String args[]){
new AEvent();
}
}
```

OUTPUT



Java MouseListener Interface

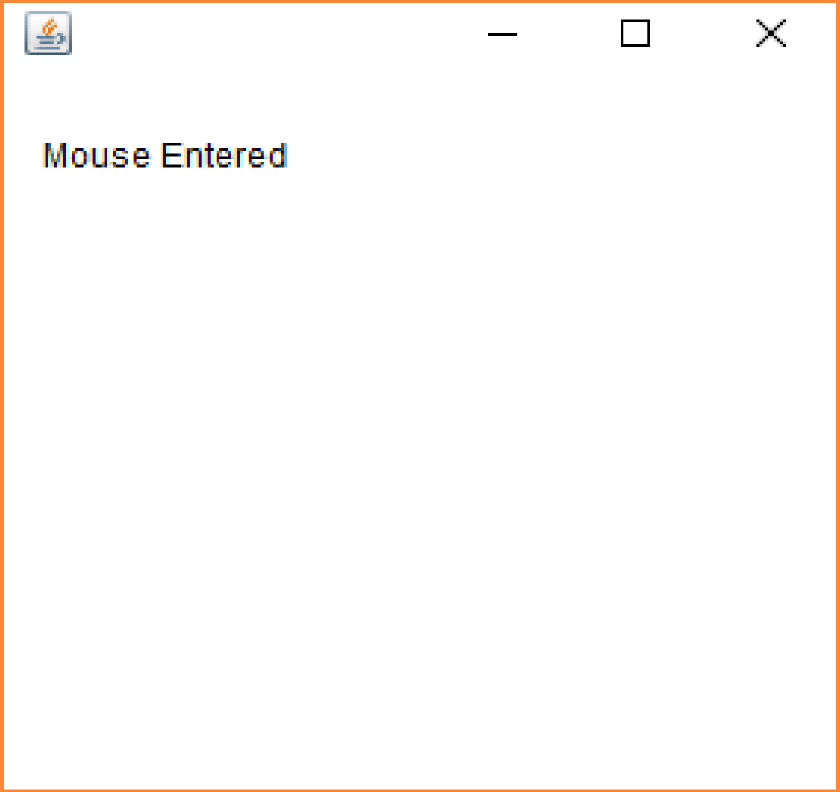
- The Java MouseListener is notified whenever you change the state of mouse.
- It is notified against MouseEvent.
- It has five methods.
- The signature of 5 methods found in MouseListener interface are given below:
  - **public abstract void** mouseClicked(MouseEvent e);
  - **public abstract void** mouseEntered(MouseEvent e);
  - **public abstract void** mouseExited(MouseEvent e);
  - **public abstract void** mousePressed(MouseEvent e);
  - **public abstract void** mouseReleased(MouseEvent e);

EXAMPLE

```
import java.awt.*;
import java.awt.event.*;
public class Example extends Frame implements
MouseListener{
    Label l;
    Example(){
        addMouseListener(this);

        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
    }
    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");
    }
    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");
    }
    public void mousePressed(MouseEvent e) {
        l.setText("Mouse Pressed");
    }
    public void mouseReleased(MouseEvent e) {
        l.setText("Mouse Released");
    }
public static void main(String[] args) {
    new Example();
} }
```

OUTPUT



```
Mouse Entered
```

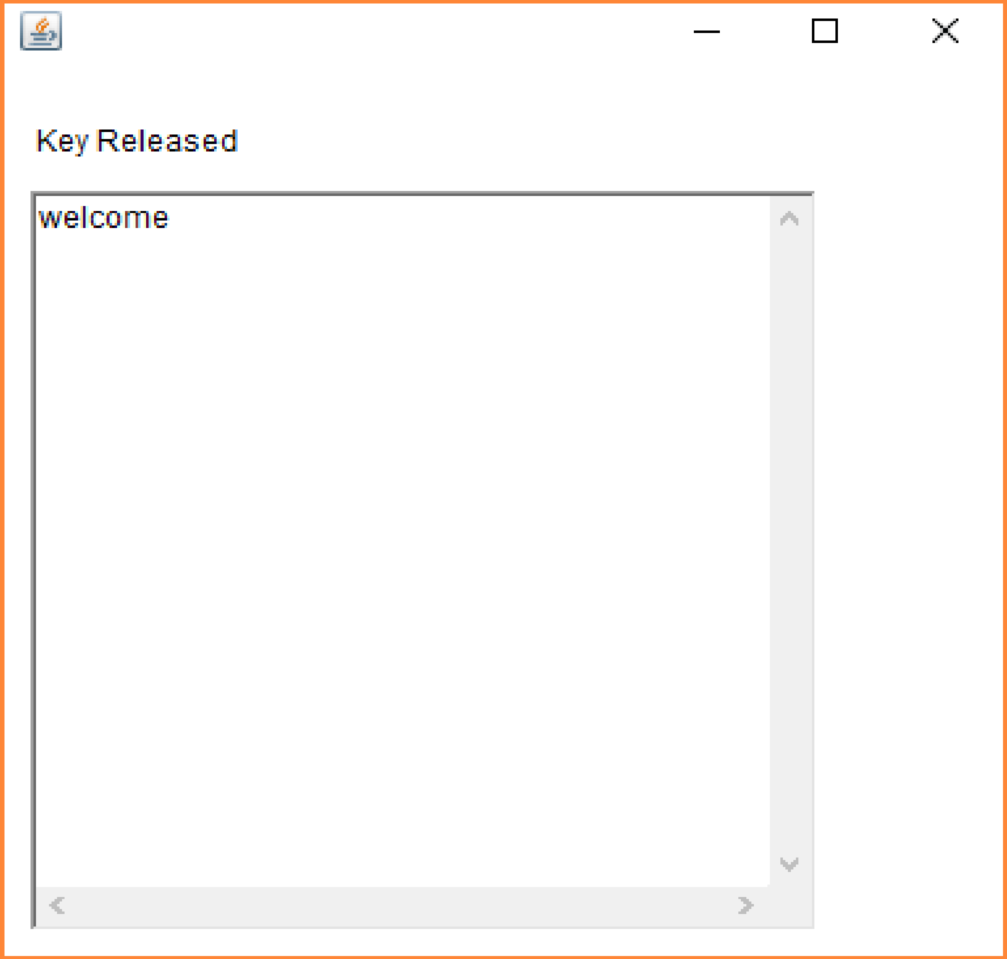
Java KeyListener Interface

- The Java KeyListener is notified whenever you change the state of key.
- It is notified against KeyEvent.
- It has three methods.
- The signature of 3 methods found in KeyListener interface are given below:
  - **public abstract void** keyPressed(KeyEvent e);
  - **public abstract void** keyReleased(KeyEvent e);
  - **public abstract void** keyTyped(KeyEvent e);

EXAMPLE

```
import java.awt.*;
import java.awt.event.*;
public class Example extends Frame implements KeyListener{
   Label l;
   TextArea area;
   Example(){
      l=new Label();
      l.setBounds(20,50,100,20);
      area=new TextArea();
      area.setBounds(20,80,300, 300);
      area.addKeyListener(this);
      add(l);add(area);
      setSize(400,400);
      setLayout(null);
      setVisible(true);
   }
   public void keyPressed(KeyEvent e) {
      l.setText("Key Pressed");
   }
   public void keyReleased(KeyEvent e) {
      l.setText("Key Released");
   }
   public void keyTyped(KeyEvent e) {
      l.setText("Key Typed");
   }
   public static void main(String[] args) {
      new Example();
   }
}
```

OUTPUT



```
Key Released

welcome
```

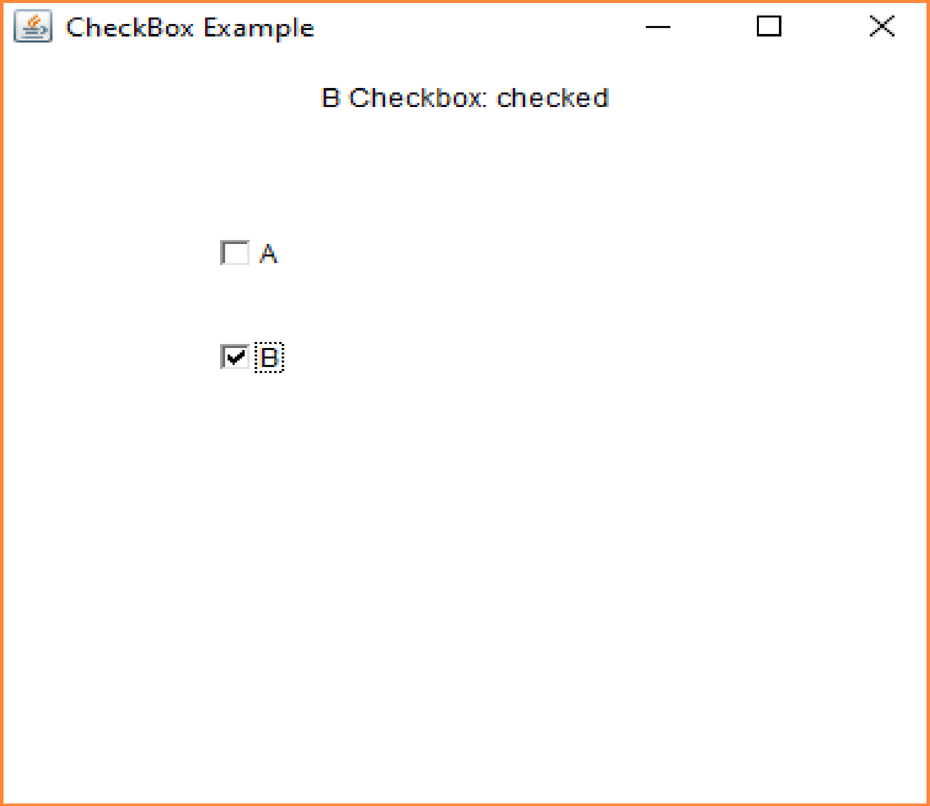
Java ItemListener Interface

- The Java ItemListener is notified whenever you click on the checkbox.
- It is notified against ItemEvent.
- It has only one method: itemStateChanged().
- The itemStateChanged() method is invoked automatically whenever you click or unclick on the registered checkbox component.
- Method signature is
  - **public abstract void** itemStateChanged(ItemEvent e);

EXAMPLE

```
import java.awt.*;
import java.awt.event.*;
public class ItemListenerExample implements ItemListener{
    Checkbox checkBox1,checkBox2;
    Label label;
    ItemListenerExample(){
        Frame f= new Frame("CheckBox Example");
        label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        checkBox1 = new Checkbox(“A");
        checkBox1.setBounds(100,100, 50,50);
        checkBox2 = new Checkbox(“B");
        checkBox2.setBounds(100,150, 50,50);
        f.add(checkBox1); f.add(checkBox2); f.add(label);
        checkBox1.addItemListener(this);
        checkBox2.addItemListener(this);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void itemStateChanged(ItemEvent e) {
      if(e.getSource()==checkBox1)
         label.setText("A Checkbox: "
         + (e.getStateChange()==1?"checked":"unchecked"));
      if(e.getSource()==checkBox2)
      label.setText("B Checkbox: "
      + (e.getStateChange()==1?"checked":"unchecked"));
    }
public static void main(String args[])
{
   new ItemListenerExample();
}
}
```

OUTPUT



```
B Checkbox: checked
```

☐ A

☑ B

Java WindowListener Interface

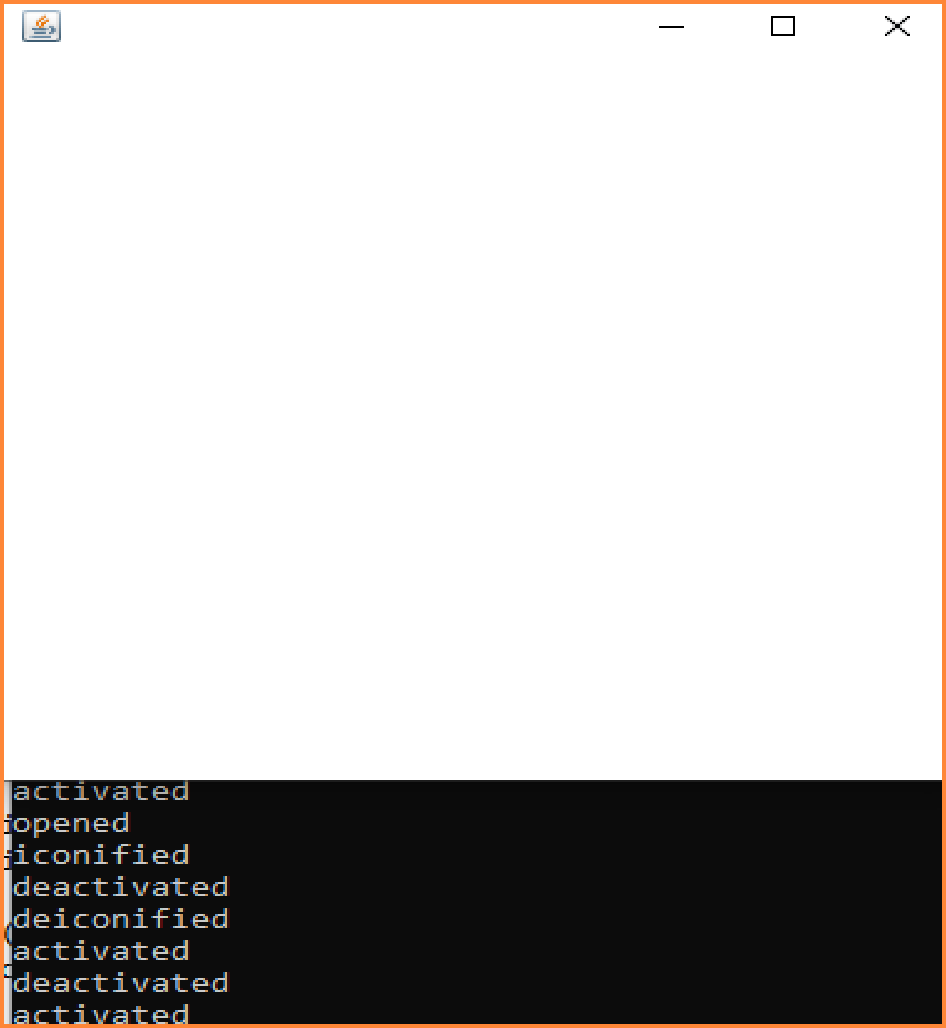
- The Java WindowListener is notified whenever you change the state of window.
- It is notified against WindowEvent.
- The signature of 7 methods found in WindowListener interface are given below:
  - **public abstract void** windowActivated(WindowEvent e);
  - **public abstract void** windowClosed(WindowEvent 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);

EXAMPLE

```
import java.awt.*;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
public class Example extends Frame implements WindowListener{
    Example(){
        addWindowListener(this);

        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
public static void main(String[] args) {
    new Example();
}
public void windowActivated(WindowEvent arg0) {
    System.out.println("activated");
}
public void windowClosed(WindowEvent arg0) {
    System.out.println("closed");
}
public void windowClosing(WindowEvent arg0) {
    System.out.println("closing");
    dispose();
}
public void windowDeactivated(WindowEvent arg0) {
    System.out.println("deactivated");
}
public void windowDeiconified(WindowEvent arg0) {
    System.out.println("deiconified");
}
public void windowIconified(WindowEvent arg0) {
    System.out.println("iconified");
}
public void windowOpened(WindowEvent arg0) {
    System.out.println("opened");
} }
```

OUTPUT
[WINDOW AND CORRESPONDING EVENT ON COMMAND PROMPT]



```
activated
opened
iconified
deactivated
deiconified
activated
deactivated
activated
```

Java Adapter Classes

- Java adapter classes *provide the* ***default implementation of listener interfaces.***
- If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces.
- So it *saves code*.
- The adapter classes are found in **java.awt.event** package.

JAVA.AWT.EVENT ADAPTER CLASSES

- The Adapter classes with their corresponding listener interfaces are given below.

Adapter class	Listener interface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

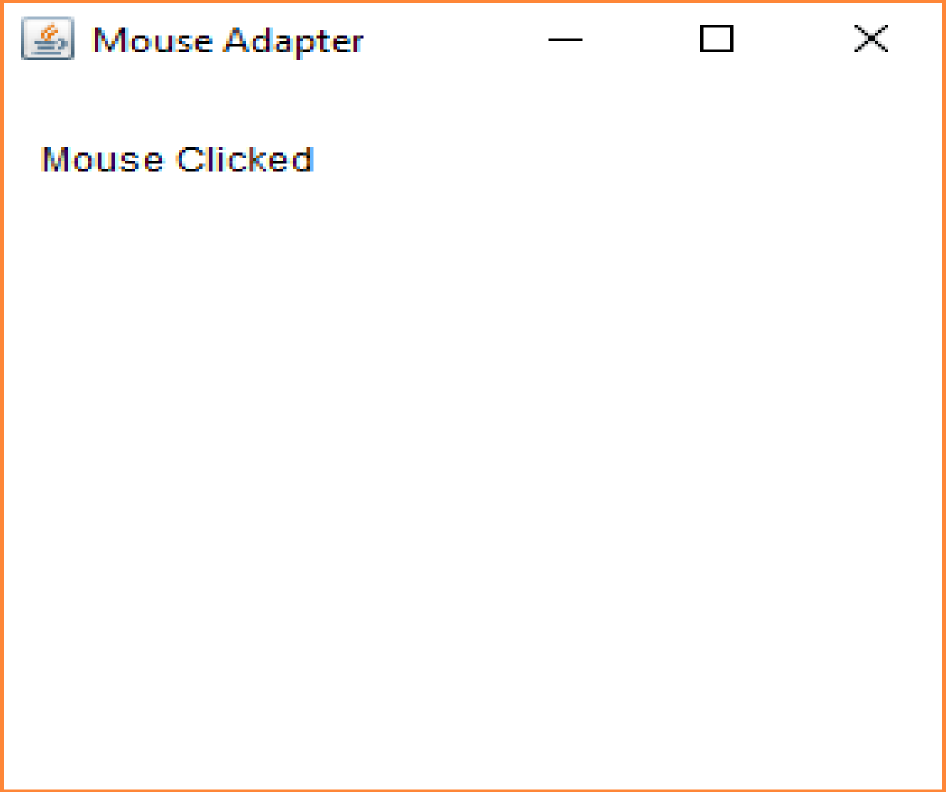
EXAMPLE

```
import java.awt.*;
import java.awt.event.*;
public class Example extends MouseAdapter{
    Frame f;
 Label l;
    Example(){
        f=new Frame("Mouse Adapter");
        f.addMouseListener(this);
         l=new Label();
        l.setBounds(20,50,100,20);
        f.add(l);

        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void mouseClicked(MouseEvent e) {
         l.setText("Mouse Clicked");
    }

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

OUTPUT



```
Mouse Clicked
```