Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)

Experiment No.1

Aim of the Experiment: Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.

Objective:

To Handle Events of AWT and Swing Components.
To Develop programs to handle events in Java Programming.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required

Course Outcome Addressed: CO2

Theory:

Event handling is fundamental to Java programming because it is used to create event driven programs eg • Applets • GUI based windows application • Web Application.

Event handling mechanism have been changed significantly between the original version of Java (1.0) and all subsequent versions of Java, beginning with version 1.1.

The modern approach to handling events is based on the delegation event model,

What is an Event?
Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

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.

What is 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. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model. The Delegation Event Model has the following key participants namely: Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provides classes for source object. Listener - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns. Advantages of

Advantages of event Handling:

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

The **Java KeyListener is notified whenever you change the state of key.** It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package, and it has three methods.

Interface declaration:

Following is the declaration for **java.awt.event.KeyListener** interface:
public interface KeyListener extends EventListener

Methods of KeyListener interface:

The signature of 3 methods found in KeyListener interface are given below:

Sr. no.	Method name	Description
1.	public abstract void keyPressed (KeyEvent e);	It is invoked when a key has been pressed.
2.	public abstract void keyReleased (KeyEvent e);	It is invoked when a key has been released.
3.	public abstract void keyTyped (KeyEvent e);	It is invoked when a key has been typed.

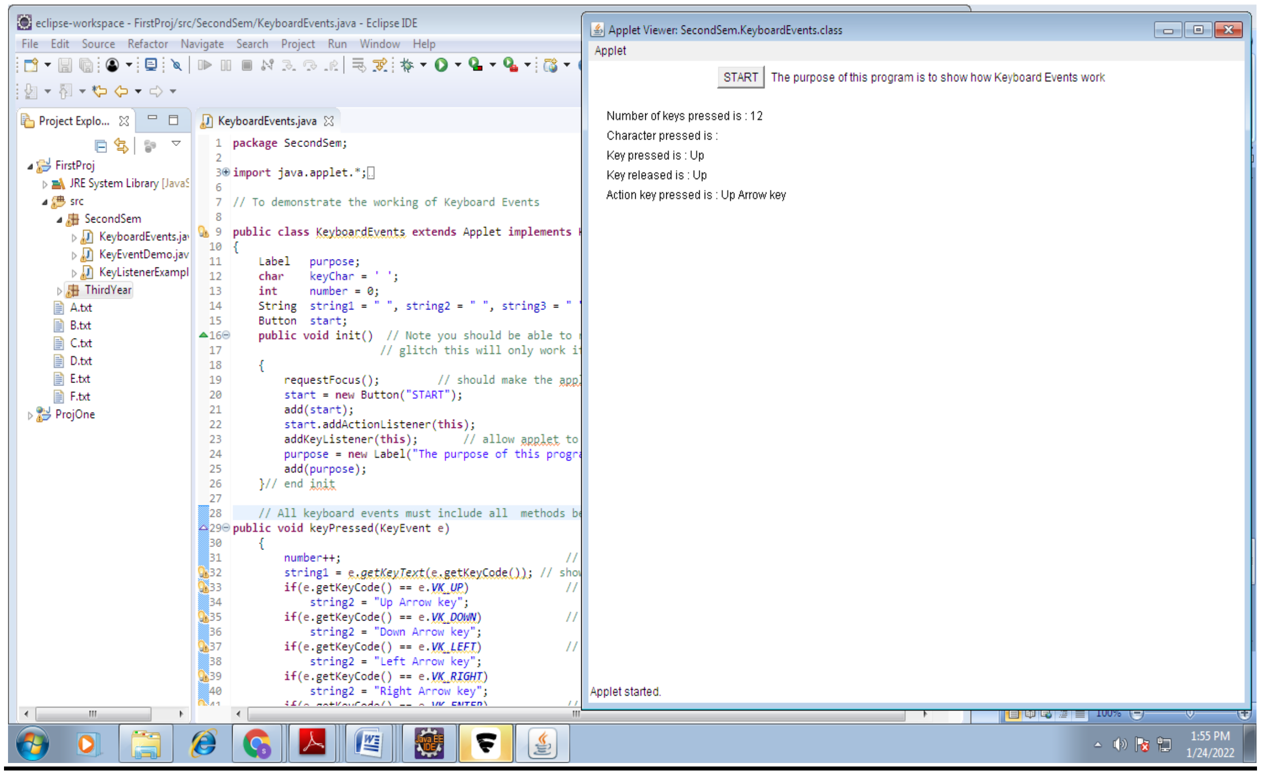
Methods inherited:

This interface inherits methods from the following interface:
java.awt.EventListener

SOURCE CODE:

```
package SecondSem;

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

// To demonstrate the working of Keyboard Events

public class KeyboardEvents extends Applet implements KeyListener , ActionListener
{
        Label    purpose;
        char     keyChar = ' ';
        int              number = 0;
        String   string1 = " ", string2 = " ", string3 = " ", string4 = " ";
        Button  start;
        public void init()  // Note you should be able to requestFocus() in init (see below) but due to a
                           // glitch this will only work if a start  button press actually activates
         requestFocus()
        {
        requestFocus();            // should make the applet the active component for Key events
        start = new Button("START");
        add(start);
        start.addActionListener(this);
        addKeyListener(this);              // allow applet to process Key events
        purpose = new Label("The purpose of this program is to show how Keyboard Events work");
        add(purpose);
        }               // end init

        // All keyboard events must include all  methods below:
public void keyPressed(KeyEvent e)
        {
        number++;                                       // add 1 to count of key presses
        string1 = e.getKeyText(e.getKeyCode()); // show which key is pressed
        if(e.getKeyCode() == e.VK_UP)                   // see if UP arrow key is pressed
        string2 = "Up Arrow key";
        if(e.getKeyCode() == e.VK_DOWN)                 // see if DOWN arrow key is pressed
        string2 = "Down Arrow key";
        if(e.getKeyCode() == e.VK_LEFT)                 // see if LEFT arrow key is pressed
        string2 = "Left Arrow key";
        if(e.getKeyCode() == e.VK_RIGHT)                // see if Right arrow key is pressed
        string2 = "Right Arrow key";
        if(e.getKeyCode() == e.VK_ENTER)             // see if ENTER key is pressed
        string2 = "Enter key";
        repaint();

        }                                       // end keyPressed

        public void keyReleased(KeyEvent e)
        {
        string4 = e.getKeyText(e.getKeyCode());
        repaint();

        }                                         // end keyReleased


        public void keyTyped(KeyEvent e)
        {
        keyChar = e.getKeyChar();            // to get the actual Unicode character typed
        if(keyChar == 'x')
        string3 = "The key lower case x was pressed";
        repaint();

        }                              // end keyTyped

        public void paint(Graphics g)
        {
        g.drawString("Number of keys pressed is : " + number, 20, 60);
        g.drawString("Character pressed is : " + keyChar, 20,80);
        g.drawString("Key pressed is : " + string1, 20, 100);
        g.drawString("Key released is : " + string4, 20, 120);
        g.drawString("Action key pressed is : " + string2, 20, 140);
        g.drawString(string3, 20, 160);

        }                                                    // end paint

        public void actionPerformed(ActionEvent e)
        {
                requestFocus();

        }                                                  // end actionPerformed

}                                                          // end KeyboardEvents
```

OUTPUT:


eclipse-workspace - FirstProj/src/SecondSem/KeyboardEvents.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explo...
KeyboardEvents.java
FirstProj
JRE System Library [JavaS
src
SecondSem
KeyboardEvents.ja
KeyEventDemo.jav
KeyListenerExampl
ThirdYear
A.txt
B.txt
C.txt
D.txt
E.txt
F.txt
ProjOne
Applet Viewer: SecondSem.KeyboardEvents.class
Applet
START
1:55 PM
1/24/2022

```
package SecondSem;

import java.applet.*;

// To demonstrate the working of Keyboard Events

public class KeyboardEvents extends Applet implements
{
    Label   purpose;
    char    keyChar = ' ';
    int     number = 0;
    String  string1 = " ", string2 = " ", string3 = "
    Button  start;
    public void init()  // Note you should be able to
                        // glitch this will only work i
    {
        requestFocus();         // should make the appl
        start = new Button("START");
        add(start);
        start.addActionListener(this);
        addKeyListener(this);       // allow applet to
        purpose = new Label("The purpose of this progra
        add(purpose);
    }// end init

    // All keyboard events must include all  methods be
public void keyPressed(KeyEvent e)
    {
        number++;                                       //
        string1 = e.getKeyText(e.getKeyCode()); // sho
        if(e.getKeyCode() == e.VK_UP)                   //
            string2 = "Up Arrow key";
        if(e.getKeyCode() == e.VK_DOWN)                 //
            string2 = "Down Arrow key";
        if(e.getKeyCode() == e.VK_LEFT)                 //
            string2 = "Left Arrow key";
        if(e.getKeyCode() == e.VK_RIGHT)
            string2 = "Right Arrow key";
```

```
The purpose of this program is to show how Keyboard Events work

Number of keys pressed is : 12
Character pressed is :
Key pressed is : Up
Key released is : Up
Action key pressed is : Up Arrow key

Applet started.
```

Conclusion:-

References:

Herbert Schildt , “Java : The Complete Reference” Tata McGraw-Hill (7th Edition).

Questions:

1.Write algorithm of a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.
2. Name any four Event Listener interfaces.
3. State the situation when all three events of KeyListener interface are generated?
4.Elaborate the terms Event, Source and Listener.
5. List various methods of ActionListener interface.