Experiment No._____ *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to demonstrate status of key on Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 1

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

Software:

1. Command prompt
2. JDK 8
3. Applet viewer

Theory:

When keyboard input is occurred, a KeyEvent is generated. There are mainly three types of key events that are recognized by integer constants as follows: KEY_PRESSED, KEY_RELEASED, KEY_TYPED When key is pressed or released, first two events are generated. When characteris pressed, the last event occurs. All Keys do not generate character. E.g. Shift Key. **InputEvent** is super class of **KeyEvent.** The class which processes the KeyEvent should implement KeyListener interface. The object of that class must be registered with a component. The object can be registered using the addKeyListener () method. Methods of KeyListener interface:

void keyPressed(KeyEvent e) : Invoked when a key is pressed

void keyReleased(KeyEvent e) : Invoked when a key has been released

void keyTyped(KeyEvent e) : Invoked when a key has been typed

Java KeyListener Interface

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

Java KeyListener Example

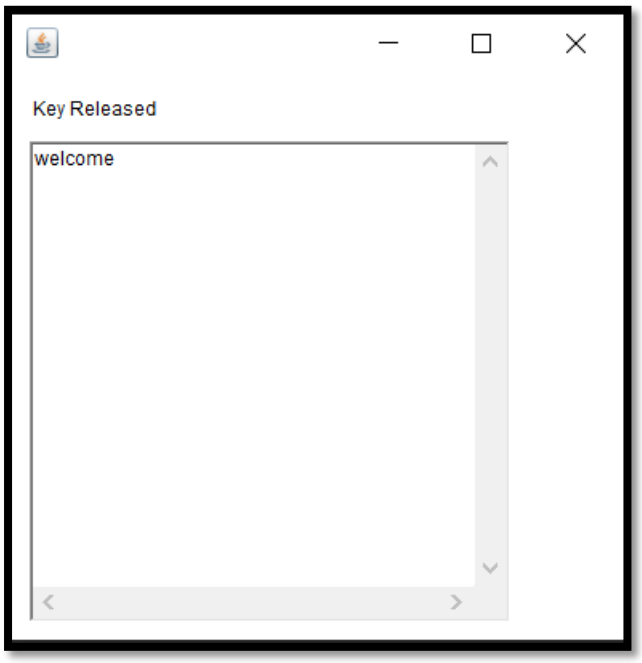
In the following example, we are implementing the methods of the KeyListener interface.

KeyListenerExample.java

```
// importing awt libraries
import java.awt.*;
import java.awt.event.*;
// class which inherits Frame class and implements KeyListener interface
public class KeyListenerExample extends Frame implements KeyListener {
// creating object of Label class   and TextArea class
 Label l;
    TextArea area;
// class constructor
   public KeyListenerExample() {
          // creating the label
        l = new Label();
// setting the location of the label in frame
        l.setBounds (20, 50, 100, 20);
// creating the text area
        area = new TextArea();
// setting the location of text area
        area.setBounds (20, 80, 300, 300);
// adding the KeyListener to the text area
        area.addKeyListener(this);
// adding the label and text area to the frame
        add(l);
add(area);
// setting the size, layout and visibility of frame
        setSize (400, 400);
        setLayout (null);
        setVisible (true);
    }
// overriding the keyPressed() method of KeyListener interface where we set the text of the label
 when key is pressed
    public void keyPressed (KeyEvent e) {
        l.setText ("Key Pressed");
    }
// overriding the keyReleased() method of KeyListener interface where we set the text of the labe
l when key is released
    public void keyReleased (KeyEvent e) {
        l.setText ("Key Released");
```

```
    }
// overriding the keyTyped() method of KeyListener interface where we set the text of the label w
hen a key is typed
    public void keyTyped (KeyEvent e) {
        l.setText ("Key Typed");
    }
  // main method
    public static void main(String[] args) {
        new KeyListenerExample();
    }
}
```

Output:



Java KeyListener Example 2: Count Words & Characters

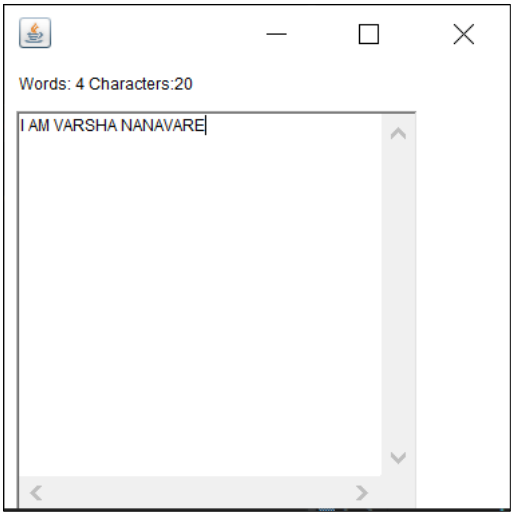
In the following example, we are printing the count of words and characters of the string. Here, the string is fetched from the TextArea and uses the KeyReleased() method of KeyListener interface.

KeyListenerExample2.java

```
// importing the necessary libraries
import java.awt.*;
import java.awt.event.*;
// class which inherits Frame class and implements KeyListener interface
public class KeyListenerExample2 extends Frame implements KeyListener {
// object of Label and TextArea
   Label l;
   TextArea area;
// class constructor
   KeyListenerExample2() {
       // creating the label
       l = new Label();
// setting the location of label
       l.setBounds (20, 50, 200, 20);
// creating the text area
       area = new TextArea();
// setting location of text area
       area.setBounds (20, 80, 300, 300);
// adding KeyListener to the text area
       area.addKeyListener(this);
 // adding label and text area to frame
  add(l);
add(area);
// setting size, layout and visibility of frame
       setSize (400, 400);
       setLayout (null);
       setVisible (true);
   }
// even if we do not define the interface methods, we need to override them
   public void keyPressed(KeyEvent e) {}
// overriding the keyReleased() method of KeyListener interface
   public void keyReleased (KeyEvent e) {
// defining a string which is fetched by the getText() method of TextArea class
       String text = area.getText();
// splitting the string in words
       String words[] = text.split ("\\s");
// printing the number of words and characters of the string
```

```
      l.setText ("Words: " + words.length + " Characters:" + text.length());
   }
   public void keyTyped(KeyEvent e) {}
  // main method
   public static void main(String[] args) {
      new KeyListenerExample2();
   }
}
```

Output:

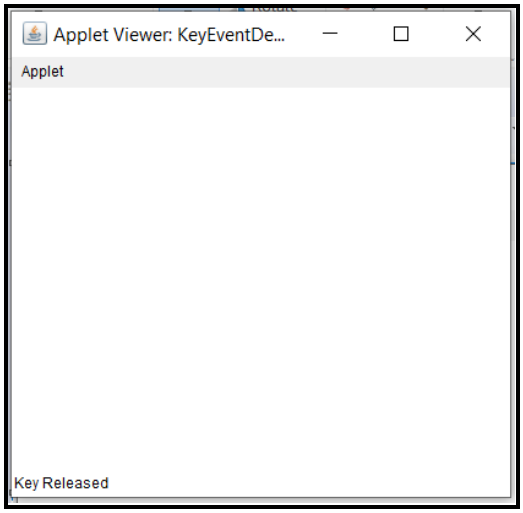


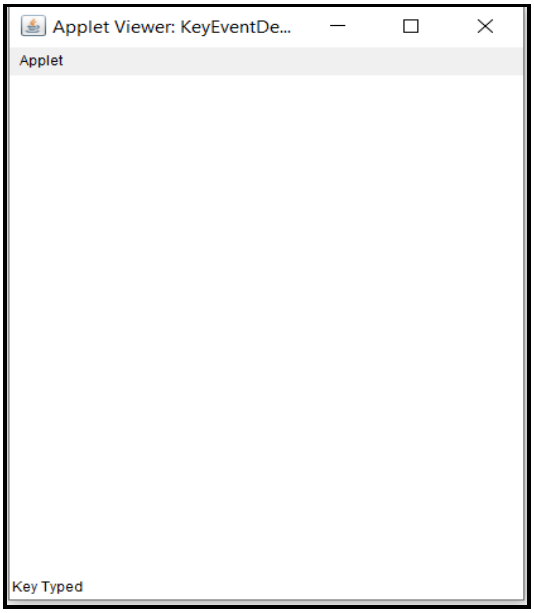
Program:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class KeyEventDemo extends Applet implements KeyListener
{
   String msg = "";

   public void init()
   {
      addKeyListener(this);
   }

   public void keyReleased(KeyEvent k)
   {
      showStatus("Key Released");
      repaint();
   }
```

```
    public void keyTyped(KeyEvent k)
    {
        showStatus("Key Typed");
        repaint();
    }

    public void keyPressed(KeyEvent k)
    {
        showStatus("Key Pressed");
        repaint();
    }

    public void paint(Graphics g)
    {
        g.drawString(msg, 10, 10);
    }
}


/*
  <applet code="KeyEventDemo" height="400" width="400">
  </applet>
*/
```

Screenshot's of Output:





Conclusion: