PRACTICAL 4

Develop an applet that displays the position of the mouse at the upper left corner of the applet when it is dragged or moved. Draw a 10x10 pixel rectangle filed with black at the current mouse position

Theory:

Java MouseMotionListener Interface:

The Java MouseMotionListener is notified whenever you move or drag mouse. It is notified against MouseEvent. The MouseMotionListener interface is found in java.awt.event package.

The signatures of 2 methods found in MouseMotionListener interface are given below:

public abstract void mouseDragged(MouseEvent e);

public abstract void mouseMoved(MouseEvent e);

Program to display mouse co-ordinates:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class GFG extends Applet implements MouseListener
{
private int x,y;
private String str = " ";
public void init()
{
this.addMouseListener (this);
//first "this" represent source (in this case it is applet which is current calling object) and
// second "this" represent listener(in this case it is GFG)
}
public void paint(Graphics g)
{  g.drawString(str,x,y);}
public void mouseClicked(MouseEvent m)
{
x = m.getX();
y = m.getY();
```

```
str = "x =" +x+",y = "+y;
repaint(); // we have made this call because repaint() will call paint() method for us.
//If we comment out this line, then we will see output only when focus is on the applet
//i.e on maximising the applet window because paint() method is called when applet screen
//gets the focus.
//repaint() is a method of Component class and prototype for this method is:
//public void repaint()
}
public void mouseEntered(MouseEvent m)
//over-riding all the methods given by
// MouseListener
{}
public void mouseExited(MouseEvent m)
{}
public void mousePressed(MouseEvent m)
{}
public void mouseReleased(MouseEvent m)
{}
}
```

Exercise:

1. Develop an applet that draws an oval filled with blue color at current mouse position.

PRACTICAL 5

Develop an applet that contains one button. Initialize the label on the button to “start”, when the user presses the button, which changes the label between these two values each time the button is pressed

Theory:

The button class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

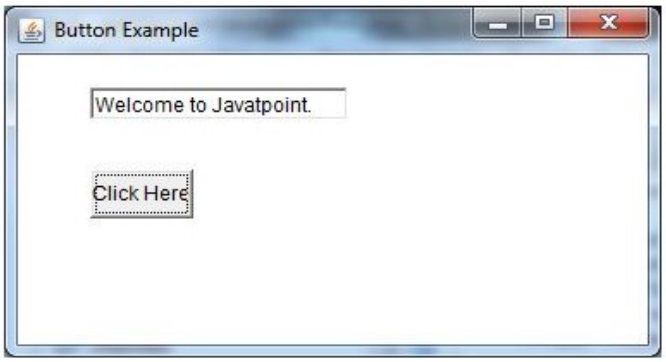
AWT Button Class declaration:

public class Button extends Component implements Accessible

A sample program to display button:

```
import java.awt.*;
import java.awt.event.*;
public class ButtonExample {
public static void main(String[] args) {
    Frame f=new Frame("Button Example");
    final TextField tf=new TextField();
    tf.setBounds(50,50, 150,20);
    Button b=new Button("Click Here");
    b.setBounds(50,100,60,30);
    b.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
            tf.setText("Welcome to Javatpoint.");
       }    });
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
} }
```

Output:



Exercise:

1. Develop an applet that contains two buttons. Initialize the label on the buttons to "red" and "green", when the user presses the button, it should change the content on the label according to the button pressed.