PROGRAMMING USING JAVA
WEEK 13 ASSIGNMENT

1. **Explain delegation event model.**

The modern approach to handling events is predicated on the delegation event model, which defines standard and consistent mechanisms to get and process events. Its concept is sort of simple: a source generates an occasion and sends it to at least one or more listeners. In this scheme, the listener simply waits until it receives an occasion. Once an occasion is received, the listener processes the event then returns.

Using the delegation event model is actually quite easy. Just follow these two steps:

1. Implement the appropriate interface in the listener so that it will receive the type of event desired.
2. Implement code to register and unregister (if necessary) the listener as a recipient for the event notifications.

Advantage of using Delegation Event Model

The advantage of this design is that the appliance logic that processes events is cleanly separated from the interface logic that 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. This provides is a crucial benefit: notifications are sent only to listeners that want to receive them. This is a more efficient way to handle events.

Note: Java also allows you to process events without using the delegation event model. This can be done by extending an AWT component.

2. Write a Java program to display the current cursor position of mouse pointer using Mouse Motion Listener interface.

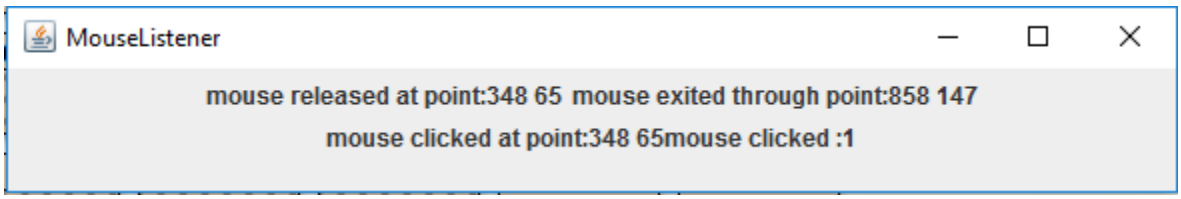
```
import java.awt.*;
import java.awt.event.*;
```

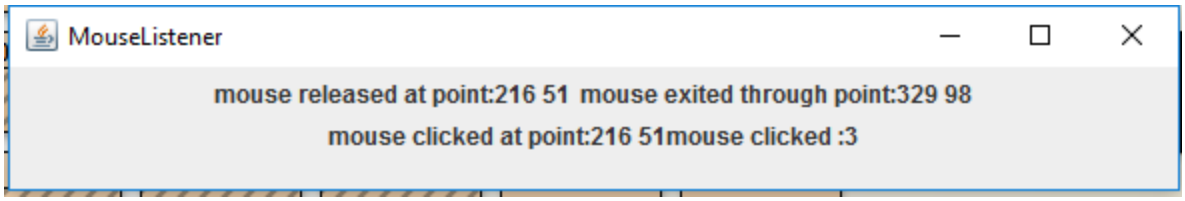
```
import javax.swing.*;
class Mouse extends Frame implements MouseListener {

    // Jlabels to display the actions of events of mouseListener //
    static JLabel label1, label2, label3;


    // default constructor
    Mouse()
    {
    }

    // main class
    public static void main(String[] args)
    {
        // create a frame
        JFrame f = new JFrame("MouseListener");

        // set the size of the frame
        f.setSize(600, 100);

        // close the frame when close button is pressed
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // create anew panel JPanel p
        = new JPanel();

        // set the layout of the panel
        p.setLayout(new FlowLayout());

        // initialize the labels
        label1 = new JLabel("no event  ");

        label2 = new JLabel("no event  "); label3

        = new JLabel("no event  ");

        // create an object of mouse class
        Mouse m = new Mouse();

        // add mouseListener to the frame
        f.addMouseListener(m);

        // add labels to the panel
        p.add(label1); p.add(label2);
        p.add(label3);
```

```
    // add panel to the frame
    f.add(p);

    f.show();
}

// getX() and getY() functions return the // x
and y coordinates of the current
// mouse position
// getClickCount() returns the number of

// quick consecutive clicks made by the user

// this function is invoked when the mouse is pressed
public void mousePressed(MouseEvent e)
{

    // show the point where the user pressed the mouse
    label1.setText("mouse pressed at point:"
                + + e.getX() + " " + e.getY());
}

// this function is invoked when the mouse is released
public void mouseReleased(MouseEvent e)
{

    // show the point where the user released the mouse click
    label1.setText("mouse released at point:"
                + + e.getX() + " " + e.getY());
}

// this function is invoked when the mouse exits the component
public void mouseExited(MouseEvent e)
{

    // show the point through which the mouse exited the frame
    label2.setText("mouse exited through point:"
                + + e.getX() + " " + e.getY());
}

// this function is invoked when the mouse enters the component
public void mouseEntered(MouseEvent e)
{

    // show the point through which the mouse entered the frame
    label2.setText("mouse entered at point:"
                + + e.getX() + " " + e.getY());
}
```

```
    // this function is invoked when the mouse is pressed or released
    public void mouseClicked(MouseEvent e)
    {
        // getClickCount gives the number of quick,
        // consecutive clicks made by the user
        // show the point where the mouse is i.e
        // the x and y coordinates
        label3.setText("mouse clicked at point:"
                    + + e.getX() + " "
                    + + e.getY() + "mouse clicked :" + e.getClickCount());
    }
}
```

Output :




MouseListener
mouse released at point:216 51 mouse exited through point:329 98
mouse clicked at point:216 51mouse clicked :3