1. Explain delegation event model.

Delegation event model is a design pattern that is used to handle events in Java. It is based on the following principles:

* An event is an object that represents a state change or an action that occurs in a source component, such as a button click or a mouse movement.
* A source component is an object that can generate events and register listeners for them, such as a button or a text field.
* A listener is an object that implements a specific interface and defines methods to handle different types of events, such as ActionListener or MouseListener.
* A handler is a method that is invoked by the listener when an event occurs, such as actionPerformed or mouseClicked.
* A delegation is a process of passing the responsibility of handling an event from the source component to the listener.

The delegation event model works as follows:

* When an event occurs in a source component, it creates an event object and passes it to the registered listeners.
* The listeners receive the event object and invoke the appropriate handler method to process the event.
* The handler method can access the information about the event and the source component from the event object and perform the desired action.

The advantages of the delegation event model are:

* It separates the logic of handling events from the logic of creating and displaying components, making the code more modular and reusable.
* It allows multiple listeners to handle the same event from the same source component, enabling flexibility and extensibility.
* It avoids inheritance and overriding of methods, which can lead to complexity and ambiguity.

1. Write a Java program to display the current cursor position of mouse pointer using Mouse Motion Listener interface.
2. // Import the Swing and AWT classes
3. import javax.swing.JFrame;
4. import javax.swing.JLabel;
5. import java.awt.BorderLayout;
6. import java.awt.event.MouseEvent;
7. import java.awt.event.MouseMotionListener;
8. // Define a class that extends JFrame and implements MouseMotionListener
9. class MouseMotionDemo extends JFrame implements MouseMotionListener {
10. // Declare a label to display the cursor position
11. private JLabel label;
12. // Define a constructor that sets up the GUI components
13. public MouseMotionDemo() {
14. // Set the title of the frame
15. setTitle("Mouse Motion Demo");
16. // Set the size of the frame
17. setSize(300, 200);
18. // Set the default close operation
19. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
20. // Create a label with some initial text
21. label = new JLabel("Move the mouse pointer over this window");
22. // Add the label to the center of the frame
23. add(label, BorderLayout.CENTER);
24. // Add this object as a mouse motion listener to the frame
25. addMouseMotionListener(this);
26. // Make the frame visible
27. setVisible(true);
28. }
29. // Override the mouseMoved method to handle mouse movement events
30. public void mouseMoved(MouseEvent e) {
31. // Get the x and y coordinates of the mouse pointer from the event object
32. int x = e.getX();
33. int y = e.getY();
34. // Set the text of the label to display the coordinates
35. label.setText("Current cursor position: (" + x + ", " + y + ")");
36. }
37. // Override the mouseDragged method to handle mouse dragging events
38. public void mouseDragged(MouseEvent e) {
39. // Do nothing
40. }
41. // Define the main method
42. public static void main(String[] args) {
43. // Create an object of MouseMotionDemo class
44. MouseMotionDemo demo = new MouseMotionDemo();
45. }
46. }