1. GUI Program to display the current mouse coordinates on the window.

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

public class MouseCoordinates extends JFrame {

    private JLabel label;

    public MouseCoordinates() {
        setTitle("Mouse Coordinates");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        label = new JLabel("Move the mouse inside the window");
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setFont(new Font("Arial", Font.PLAIN, 16));
        add(label, BorderLayout.SOUTH);

        // Add mouse motion listener to the content pane
        getContentPane().addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                label.setText("Mouse Coordinates: X = " + e.getX() + ", Y = " + e.getY());
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(MouseCoordinates::new);
    }
}
```

2. GUI Program to implement a simple Timer (using background events). Include a Start and Stop button to control the timer.

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

public class SimpleTimer extends JFrame {

    private JLabel timerLabel;
    private JButton startButton, stopButton;
    private Timer timer;
    private int seconds = 0;

    public SimpleTimer() {
        setTitle("Simple Timer");
        setSize(300, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        timerLabel = new JLabel("Time: 0 seconds");
        timerLabel.setFont(new Font("Arial", Font.BOLD, 16));
        add(timerLabel);

        startButton = new JButton("Start");
        stopButton = new JButton("Stop");

        add(startButton);
        add(stopButton);

        // Timer that updates every 1 second (1000 ms)
        timer = new Timer(1000, e -> {
            seconds++;
            timerLabel.setText("Time: " + seconds + " seconds")
        });

        // Start button action
        startButton.addActionListener(e -> {
            if (!timer.isRunning()) {
                timer.start();
            }
        });

        // Stop button action
        stopButton.addActionListener(e -> {
            if (timer.isRunning()) {
                timer.stop();
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(SimpleTimer::new);
    }
}
```

3. Create a GUI with a JComboBox containing image names. On selection, display the corresponding image using a JLabel and ItemListener.

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

public class ImageSelector extends JFrame {

    private JComboBox<String> imageComboBox;
    private JLabel imageLabel;

    // List of image names (assumes images are in the same directory or classpath)
    private String[] imageNames = { "cat.jpeg", "dog.jpeg", "bird.jpeg" };

    public ImageSelector() {
        setTitle("Image Selector");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        imageComboBox = new JComboBox<>(imageNames);
        imageLabel = new JLabel("", SwingConstants.CENTER);

        // Add ItemListener to ComboBox
        imageComboBox.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    String selectedImage = (String) imageComboBox.getSelectedItem();
                    ImageIcon icon = new ImageIcon(selectedImage);
                    Image img = icon.getImage().getScaledInstance(300, 300, Image.SCALE_SMOOTH);
                    imageLabel.setIcon(new ImageIcon(img));
                }
            }
        });

        add(imageComboBox, BorderLayout.NORTH);
        add(imageLabel, BorderLayout.CENTER);

        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ImageSelector::new);
    }
}
```

4. GUI with a JTextArea and a label. As the user types, show the character count and word count in real-time using a KeyListener.

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

public class TextCounter extends JFrame {

    private JTextArea textArea;
    private JLabel statusLabel;

    public TextCounter() {
        setTitle("Live Text Counter");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        textArea = new JTextArea();
        textArea.setLineWrap(true);
        textArea.setFont(new Font("Arial", Font.PLAIN, 16));
        JScrollPane scrollPane = new JScrollPane(textArea);
        add(scrollPane, BorderLayout.CENTER);

        statusLabel = new JLabel("Characters: 0 | Words: 0");
        statusLabel.setFont(new Font("Arial", Font.BOLD, 14));
        add(statusLabel, BorderLayout.SOUTH);

        // KeyListener to track changes
        textArea.addKeyListener(new KeyAdapter() {
            @Override
            public void keyReleased(KeyEvent e) {
                updateCounts();
            }
        });

        setVisible(true);
    }

    private void updateCounts() {
        String text = textArea.getText();
        int charCount = text.length();

        // Split by whitespace and filter out empty strings
        String[] words = text.trim().split("\\s+");
        int wordCount = text.trim().isEmpty() ? 0 : words.length;

        statusLabel.setText("Characters: " + charCount + " | Words: " + wordCount);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(TextCounter::new);
    }
}
```

5. Write Java GUI Program using Swing to change background on selecting color.

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

public class ColorChanger extends JFrame {

    private JComboBox<String> colorComboBox;
    private JPanel mainPanel;

    // Color names and their corresponding Color objects
    private final String[] colorNames = { "White", "Red", "Green", "Blue", "Yellow", "Orange", "Gray", "Cyan" };

    public ColorChanger() {
        setTitle("Background Color Changer");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Panel to change background color
        mainPanel = new JPanel();
        mainPanel.setLayout(new FlowLayout());

        // Combo box for color selection
        colorComboBox = new JComboBox<>(colorNames);
        mainPanel.add(new JLabel("Select a color:"));
        mainPanel.add(colorComboBox);

        // Item listener to change background color
        colorComboBox.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    String selectedColor = (String) colorComboBox.getSelectedItem();
                    mainPanel.setBackground(getColorByName(selectedColor));
                }
            }
        });

        add(mainPanel);
        setVisible(true);
    }

    // Utility method to convert color name to Color object
    private Color getColorByName(String name) {
        return switch (name) {
            case "Red" -> Color.RED;
            case "Green" -> Color.GREEN;
            case "Blue" -> Color.BLUE;
            case "Yellow" -> Color.YELLOW;
            case "Orange" -> Color.ORANGE;
            case "Gray" -> Color.GRAY;
            case "Cyan" -> Color.CYAN;
            default -> Color.WHITE;
        };
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ColorChanger::new);
    }
}
```