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

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MouseLocation
extends JFrame implements
MouseMotionListener {
   JLabel label;
   public MouseLocation() {
      setTitle("Mouse Coordinate
Tracker");
      setSize(400, 300);
      setLayout(null);

setDefaultCloseOperation(EXIT_O
N_CLOSE);
      label = new JLabel("Move the
mouse");
      label.setBounds(100, 100,
200, 30);
      label.setFont(new
Font("Arial", Font.PLAIN, 16));
      add(label);

addMouseMotionListener(this);
      setVisible(true);
   }
   public void
mouseMoved(MouseEvent e) {
      int x = e.getX();
      int y = e.getY();
      label.setText("Mouse at: (" +
x + ", " + y + ")");
   }
   public void
mouseDragged(MouseEvent e) {
      // Optional: you can update
coordinates while dragging if
needed
      mouseMoved(e);
   }
   public static void main(String[]
args) {
      new MouseLocation();
   }
}
```

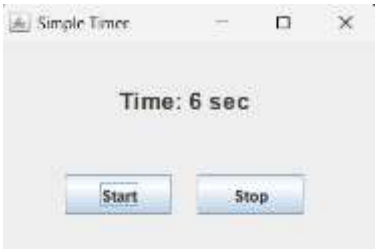
OUTPUT

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 implements ActionListener
{
   JLabel timeLabel;
   JButton startBtn, stopBtn;
   Timer timer;
   int seconds = 0;
   public SimpleTimer() {
      setTitle("Simple Timer");
      setSize(300, 200);
      setLayout(null);

setDefaultCloseOperation(EXIT_O
N_CLOSE);
      timeLabel = new
JLabel("Time: 0 sec");
      timeLabel.setBounds(90, 30,
150, 30);
      timeLabel.setFont(new
Font("Arial", Font.BOLD, 18));
      add(timeLabel);
      startBtn = new
JButton("Start");
```

```
        startBtn.setBounds(50, 100,
80, 30);
        add(startBtn);
        stopBtn = new
JButton("Stop");
        stopBtn.setBounds(150, 100,
80, 30);
        add(stopBtn);

startBtn.addActionListener(this);

stopBtn.addActionListener(this);
        // Timer: fires every 1000 ms
(1 sec), increments seconds
        timer = new Timer(1000, new
ActionListener() {
            public void
actionPerformed(ActionEvent e) {
                seconds++;
                timeLabel.setText("Time:
" + seconds + " sec");
            }
        });
        setVisible(true);
    }
    public void
actionPerformed(ActionEvent e) {
        if (e.getSource() == startBtn)
{
            timer.start();
        } else if (e.getSource() ==
stopBtn) {
            timer.stop();
        }
    }
   public static void main(String[]
args) {
        new SimpleTimer();
    }
}
```

output

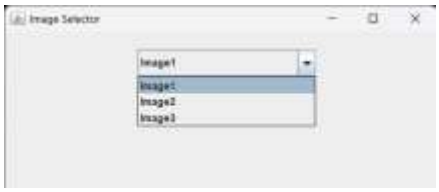


3. Create a GUI with a JComboBox containing image names. On selection, display the corresponding image using a JLabel and ItemListener.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ImageSelector extends
JFrame implements ItemListener {
    JComboBox<String> imageList;
    JLabel imageLabel;
    String[] imageNames =
{"Image1", "Image2", "Image3"};
    String[] imagePaths =
{"image1.jpeg", "image2.jpeg",
"image3.jpeg"};
    public ImageSelector() {
        setTitle("Image Selector");
        setSize(500, 400);
        setLayout(null);
setDefaultCloseOperation(EXIT_O
N_CLOSE);
        imageList = new
JComboBox<>(imageNames);
        imageList.setBounds(150, 20,
200, 30);
        add(imageList);
        imageLabel = new JLabel();
        imageLabel.setBounds(100,
70, 300, 250);
imageLabel.setHorizontalAlignme
nt(JLabel.CENTER);
        add(imageLabel);


imageList.addItemListener(this);
        setVisible(true);
```

```
    }
    public void
itemStateChanged(ItemEvent e) {
        if (e.getStateChange() ==
ItemEvent.SELECTED) {
            int index =
imageList.getSelectedIndex();
            ImageIcon icon = new
ImageIcon(imagePaths[index]);
            Image img =
icon.getImage().getScaledInstance(
300, 250,
Image.SCALE_SMOOTH);
            icon = new
ImageIcon(img);
            imageLabel.setIcon(icon);
        }
    }
    public static void main(String[]
args) {
        new ImageSelector();
    }
}
```

output



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 TextCounterGUI
extends JFrame implements
KeyListener {
    private JTextArea textArea;
    private JLabel countLabel;

    public TextCounterGUI() {
        setTitle("Live Text Counter");
        setSize(400, 300);
setDefaultCloseOperation(JFrame.
EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        textArea = new JTextArea();
        countLabel = new
JLabel("Characters: 0 | Words: 0");

textArea.addKeyListener(this);
        setLayout(new
BorderLayout());
        add(new
JScrollPane(textArea),
BorderLayout.CENTER);
        add(countLabel,
BorderLayout.SOUTH);
setVisible(true);
    }
    public void
keyPressed(KeyEvent e) {
        // Not used, but must be
implemented
    }
    public void
keyReleased(KeyEvent e) {
        updateCounts();
    }
    public void keyTyped(KeyEvent
e) {
        // Not used, but must be
implemented
    }
private void updateCounts() {
        String text =
textArea.getText();
        int charCount = text.length();
        int wordCount =
text.trim().isEmpty() ? 0 :
text.trim().split("\\s+").length;
```

```
countLabel.setText("Characters: "
+ charCount + " | Words: " +
wordCount);
    }
    public static void main(String[]
args) {
SwingUtilities.invokeLater(TextCo
unterGUI::new);
    }
}
```

Output

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

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

class bgcolor extends JFrame
implements ActionListener{
JButton B1,B2,B3;
public bgcolor(){
B1=new JButton("Red");
B2=new JButton("Green");
B3=new JButton("Blue");
B1.setBounds(20,20,80,30);
B2.setBounds(120,20,80,30);
B3.setBounds(220,20,80,30);
add(B1);
add(B2);
add(B3);
B1.addActionListener(this);
B2.addActionListener(this);
B3.addActionListener(this);
setLayout(null);
setSize(400,400);
setVisible(true);
setDefaultCloseOperation(JFrame.
EXIT_ON_CLOSE);
}
public void
actionPerformed(ActionEvent e)
{
if(e.getSource()==B1){
getContentPane().setBackground(C
olor.RED);
}
else if (e.getSource()==B2){
getContentPane().setBackground(C
olor.GREEN);
}else
{
getContentPane().setBackground(C
olor.BLUE);
}
}
public static void main(String
args[])
{
bgcolor r=new bgcolor();
}
}
```

Output