Q.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 coordinatesLabel;

public MouseCoordinates() {

setTitle("Mouse Coordinates Display");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

coordinatesLabel = new JLabel("Move the mouse inside the window"); coordinatesLabel.setHorizontalAlignment(SwingConstants.CENTER);

coordinatesLabel.setFont(new Font("Arial", Font.PLAIN, 16));

add(coordinatesLabel, BorderLayout.SOUTH);

addMouseMotionListener(new MouseMotionAdapter() {

@Override

public void mouseMoved(MouseEvent e) {

coordinatesLabel.setText("Mouse Coordinates: X = " + e.getX() + ", Y = " + e.getY());

}

});

setVisible(true);

}

public static void main(String[] args) {

new MouseCoordinates();

}

}

Q.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 timeLabel;

private JButton startButton, stopButton;

private Timer timer;

private int elapsedTime = 0;

public SimpleTimer() {

setTitle("Simple Timer");

setSize(300, 150); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new FlowLayout());

timeLabel = new JLabel("Elapsed Time: 0 seconds");

timeLabel.setFont(new Font("Arial", Font.BOLD, 16));

startButton = new JButton("Start");

stopButton = new JButton("Stop");

add(timeLabel);

add(startButton);

add(stopButton);

timer = new Timer(1000, new ActionListener() {

public void actionPerformed(ActionEvent e) {

elapsedTime++;

timeLabel.setText("Elapsed Time: " + elapsedTime + " seconds");

}

});

startButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

timer.start();

}

});

stopButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

timer.stop();

}

});

setVisible(true);

}

public static void main(String[] args) {

new SimpleTimer();

}

}

Q. 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;

private String[] imageNames = {"cat.jpg", "dog.jpg", "bird.jpg"};

public ImageSelector() {

setTitle("Image Viewer");

setSize(400, 400); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

imageComboBox = new JComboBox<>(imageNames);

imageLabel = new JLabel("", SwingConstants.CENTER);

add(imageComboBox, BorderLayout.NORTH);

add(imageLabel, BorderLayout.CENTER);

imageComboBox.addItemListener(new ItemListener() {

public void itemStateChanged(ItemEvent e) {

if (e.getStateChange() == ItemEvent.SELECTED) {

String selectedImage = (String) imageComboBox.getSelectedItem();

ImageIcon icon = new ImageIcon(selectedImage);

Image scaledImage = icon.getImage().getScaledInstance(300, 300, Image.SCALE\_SMOOTH);

imageLabel.setIcon(new ImageIcon(scaledImage));

}

}

});

setVisible(true);

}

public static void main(String[] args) {

new ImageSelector();

}

}

Q.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 TextAnalyzer extends JFrame {

private JTextArea textArea;

private JLabel countLabel;

public TextAnalyzer() {

setTitle("Real-Time Text Analyzer");

setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

textArea = new JTextArea();

countLabel = new JLabel("Characters: 0 | Words: 0");

textArea.setLineWrap(true);

textArea.setWrapStyleWord(true);

add(new JScrollPane(textArea), BorderLayout.CENTER);

add(countLabel, BorderLayout.SOUTH);

textArea.addKeyListener(new KeyAdapter() {

public void keyReleased(KeyEvent e) {

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);

}

});

setVisible(true);

}

public static void main(String[] args) {

new TextAnalyzer();

}

}

Q. 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 BackgroundColorChanger extends JFrame {

private JComboBox<String> colorComboBox;

private JPanel mainPanel;

public BackgroundColorChanger() {

setTitle("Background Color Changer");

setSize(400, 300); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

String[] colors = {"White", "Red", "Green", "Blue", "Yellow", "Pink", "Gray"};

colorComboBox = new JComboBox<>(colors);

mainPanel = new JPanel();

mainPanel.setLayout(new BorderLayout());

mainPanel.add(colorComboBox, BorderLayout.NORTH);

add(mainPanel);

colorComboBox.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String selectedColor = (String) colorComboBox.getSelectedItem();

switch (selectedColor) {

case "Red": mainPanel.setBackground(Color.RED);

break;

case "Green": mainPanel.setBackground(Color.GREEN);

break;

default: mainPanel.setBackground(Color.WHITE);

}

}

});

setVisible(true);

}

public static void main(String[] args) {

new BackgroundColorChanger();

}}