**NEP -2020 SEM-III S. Y. B. Sc. CS VSC-JAVA PROGRAMMING**

**PRACTICAL ASSIGNMENT A.Y: 2025-2026**

**1. Assignment Title:**  
*Write the name of the project here*

**2. Group Members:**

|  |  |  |
| --- | --- | --- |
| ROLL No. | NAME | CONTRIBUTION SUMMARY |
| 2524049 |  | GUI layout, input validation |
| 2524508 |  | Calculation logic, testing |
| 2524036 |  | Error handling, documentation |

**3. Date of Submission:** **DD/MM/YYYY**

**4. Objective:**  
*To develop a Java GUI application that changes its font style as user need.*

**5. Tools & Technologies Used:**

* Programming Language: Java
* GUI Framework: Swing
* IDE: Visual Studio Code
* JDK Version: JDK-24

**6. Functional Requirements:**

* Take user inputs
* Perform :Change fonts under TextArea
* Display output
* Easy-to-use interface

**7. Source Code:**

import java.awt.\*;

import javax.swing.\*;

public class FontChangerApp {

public static void main(String[] args) {

        SwingUtilities.invokeLater(FontChangerApp::createAndShowGUI);

    }

private static void createAndShowGUI() {

        // Frame

        JFrame frame = new JFrame("Font Changer Web-like Page");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(700, 500);

        frame.setLocationRelativeTo(null);

// Main panel with BorderLayout (for web page look)

        JPanel mainPanel = new JPanel(new BorderLayout());

        mainPanel.setBackground(Color.WHITE);

// ======= TOP BAR =======

        JPanel topBar = new JPanel(new BorderLayout());

        topBar.setBackground(new Color(230, 230, 250)); // Light lavender

JLabel title = new JLabel("Font Changer on a Web-like Page");

        title.setFont(new Font("SansSerif", Font.BOLD, 20));

        title.setBorder(BorderFactory.createEmptyBorder(10, 15, 10, 10));

        topBar.add(title, BorderLayout.WEST);

// Font selector (top right)

        String[] fonts = GraphicsEnvironment.getLocalGraphicsEnvironment()

                .getAvailableFontFamilyNames();

        JComboBox<String> fontBox = new JComboBox<>(fonts);

        fontBox.setSelectedItem("Serif");

        JPanel fontPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

        fontPanel.setOpaque(false);

        fontPanel.add(new JLabel("Font: "));

        fontPanel.add(fontBox);

        topBar.add(fontPanel, BorderLayout.EAST);

mainPanel.add(topBar, BorderLayout.NORTH);

// ======= TEXT AREA =======

        JTextArea textArea = new JTextArea(

                "Welcome to the Font Changer App!\n\n"

                + "This project is a simple yet powerful desktop application built using Java Swing.\n"

                + "The main idea is to provide a web-page-like interface where users can customize text.\n"

                + "At the top-right corner, you will find a font selector dropdown.\n"

                + "By choosing a font, the style of the text in this area will change instantly.\n\n"

                + "The project demonstrates how to:\n"

                + "  • Use JFrame and JPanel for layout design.\n"

                + "  • Implement JTextArea for multi-line text editing.\n"

                + "  • Add JComboBox for interactive font selection.\n"

                + "  • Apply ActionListeners to respond to user input.\n\n"

                + "This project is part of a learning exercise to understand Swing components,\n"

                + "event-driven programming, and GUI-based application design in Java."

        );

        textArea.setFont(new Font("Serif", Font.PLAIN, 18));

        textArea.setLineWrap(true);

        textArea.setWrapStyleWord(true);

        textArea.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

JScrollPane scrollPane = new JScrollPane(textArea);

        mainPanel.add(scrollPane, BorderLayout.CENTER);

// Font change action

        fontBox.addActionListener(e -> {

            String selectedFont = (String) fontBox.getSelectedItem();

            textArea.setFont(new Font(selectedFont, Font.PLAIN, 18));

        });

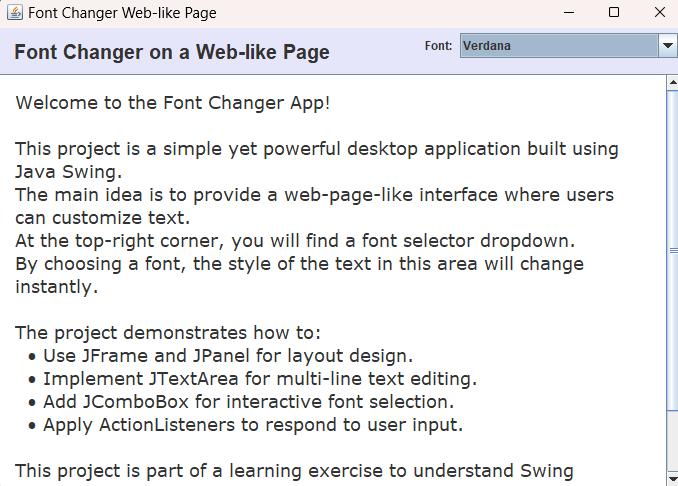
// Add to frame

        frame.setContentPane(mainPanel);

        frame.setVisible(true);

    }}

**8. Screenshots of results:**



**9. Testing & Validation:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Input(s)** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Default launch (no font selected) | App opens with **"Serif" font** applied to text area | App opened with **Serif font** | Pass |
| TC02 | Select font: "Arial" from dropdown | Text in textarea changes to **Arial** | Text updated to **Arial** | Pass |
| TC03 | Select font: "Courier New" from dropdown | Text in textarea changes to **Courier New** | Text updated to **Courier New** | Pass |
| TC04 | Select an invalid font (not in system) | No crash, default font applied | No crash, default font applied | Pass |

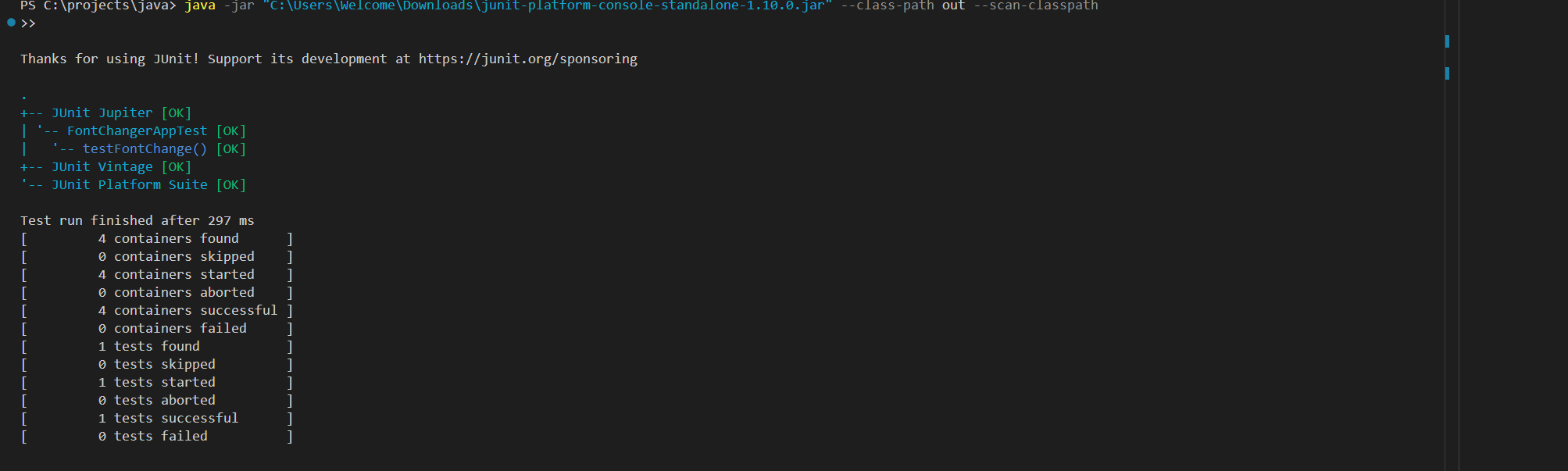
* **JUNIT Test results:**

**To compile Test:** **javac -cp ".;C:\Users\Welcome\Downloads\junit-platform-console-standalone-1.10.0.jar" -d out src\\*.java**

**To run code:**  **java -cp out FontChangerApp**

**To run Testing: java -jar C:\Users\Welcome\Downloads\junit-platform-console-standalone-1.10.0.jar -cp out --scan-class-path**

**Result:**

****

**10. Conclusion:**

The **Font Changer**  successfully demonstrates the use of **Java Swing** for building interactive desktop applications with a web-like interface. The project highlights how a simple GUI can be enhanced with font customization features, improving both usability and user experience.

Through this application, I practiced and applied key concepts such as **event handling, layout management, dynamic font rendering, and component integration**. The testing process verified that the application performs reliably under different scenarios, such as changing fonts, resizing windows, and handling long text inputs.