SCE SUBMISSION

Object Oriented Programming

[PATTERN 2020]

Second Year Bachelor Of Technology

AY-2023-24, Sem-IV

Course Teacher

Mrs. Dipti Pandit

Head of Department

Prof. (Dr.) Shraddha Habbu

Department of Electronics and Telecommunication Engineering

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

____

Mrs .Dipti Pandit

(Course Teacher)

Group Members			
Sr.No	**Roll Number**	**Prn Number**	**Name**
1	212076	22320181	Mahamune Abhijeet Pandurang
2	212069	22320053	Bhangare Harshal Pramod
3	212067	22320022	Mahamuni Chitanya Santosh
4	212055	22210945	Sarvade Akanksha Balasaheb

Index:

1. Introduction

The Advanced Notepad project aims to develop a versatile text editing application using Java programming language. This notepad application provides users with an intuitive graphical user interface (GUI) for creating, editing, saving, and opening text documents. The project leverages Java's Swing library for building the GUI components and implements various functionalities to enhance user experience.

2. Objectives

The primary objectives of the Advanced Notepad project are as follows:

- **User-Friendly Interface:** Create a simple and intuitive user interface for seamless text editing experience.
- **Basic Functionalities:** Implement fundamental text editing functionalities such as opening, saving, and editing text documents.
- **Advanced Features:** Enhance the application with advanced features including cut, copy, paste, undo, redo, find and replace, and formatting options.
- **File Handling:** Support file handling operations such as opening and saving text files from and to the local filesystem.
- **Error Handling:** Implement robust error handling mechanisms to provide smooth execution and user feedback.

3. Technologies Used

- **The Advanced Notepad project utilizes the following technologiesJava SE Development Kit (JDK):** Version 8 or higher is required to develop and run the Java application.
- **Integrated Development Environment (IDE):** IntelliJ IDEA or Eclipse IDE is recommended for Java development.

4. Project Structure

The project follows a standard packages used :

- Java.awt
- Java.io
- Javax.swing
- The Advanced Notepad application utilizes Swing GUI components to create the user interface:

- JTextArea: Used for displaying and editing text content.
- JMenuBar, JMenu, JMenuItem: Components for creating the menu bar and dropdown menus.
- JFileChooser: Dialog for selecting files to open or save.

5 Basic Functionalities

- The application implements basic text editing functionalities:
- Open: Allows users to open existing text files.
- Save: Enables users to save the current document to a file.
- Edit: Provides standard text editing operations such as cut, copy, paste, undo, and redo.
- Advanced Features
- Advanced features are implemented to enhance user productivity:
- Find and Replace: Allows users to search for specific text and replace it with another.
- Font Selection: Provides options for changing the font family, size, and style of the text.
- Text Formatting: Supports basic text formatting options such as bold, italic, and underline.
- File Handling
- The application supports file handling operations:
- Open File: Users can open existing text files from their local filesystem.
- Save File: Users can save the current document to a file with a specified name and location.
- Error Handling
- Robust error handling mechanisms are implemented to ensure smooth execution:
- File I/O Errors: Handles errors related to file input/output operations.
- User Input Validation: Validates user input to prevent unexpected behavior.

6. Future Enhancements

Several enhancements can be considered to improve the Advanced Notepad application:

Cloud Integration: Integration with cloud storage services for seamless file synchronization and backup.

Tabbed Interface: Support for multiple tabs for editing multiple documents simultaneously.

Syntax Highlighting: Implementation of syntax highlighting for different programming languages.

Version Control Integration: Integration with version control systems (e.g., Git) for collaborative editing and version history.

Customizable Themes: Provide users with customizable themes and user preferences for the application.

7 Data Structure and OOP Concept Analysis

1. Data Structures

a. Text Storage

Data Structure:

The primary data structure used for storing the text content in the notepad is typically a dynamic data structure, such as a resizable array or a linked list.

Usage: This data structure allows for efficient storage and manipulation of text data, enabling features like editing, inserting, and deleting characters or lines.

b. Recent Files List

Data Structure: To maintain the list of recently opened files, a suitable data structure like a linked list or an array may be used.

Usage: This data structure facilitates easy access to the most recently opened files, allowing users to quickly revisit them without navigating the file system.

c. Undo/Redo History

Data Structure: A stack-based data structure is commonly used to implement the undo/redo functionality.

Usage: Each editing action performed by the user, such as inserting or deleting text, is pushed onto the undo stack. When the user invokes the undo command, the most recent action is popped from the undo stack and pushed onto the redo stack.

2. Object-Oriented Programming (OOP) Concepts

a. Encapsulation

Usage: OOP principles are employed to encapsulate related functionalities into classes, ensuring data hiding and modularity. For instance, the Notepad class may encapsulate the entire notepad application, providing methods to interact with the text editor, file operations, and user interface components.

b. Inheritance

Usage: Inheritance may be utilized to create specialized classes that extend the functionality of existing classes. For example, a SyntaxHighlightingTextPane class may inherit from a standard text pane class to incorporate syntax highlighting features specific to programming languages.

c. Polymorphism

Usage: Polymorphism allows for the flexibility of operations to work on objects of different types. For instance, various text formatting options, such as bold, italic, or underline, may be implemented using polymorphism, where a common interface or abstract class defines the behavior for applying formatting to text.

8 Implementation code with result

```
// package Abhijeet;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
```

```java
import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;


public class Anotepad extends JFrame implements ActionListener, KeyListener {

   // initializing

   int fontsize1 = 20;

   JLabel labelword, labelfsize, textLength;

   JTextArea ta;

   JMenuBar menubar;

   JMenu menu1, menu2, menu3;

   JMenuItem item1, item2, item3, item4, item5,item6, open, fontsize, cut, copy, paste,
selectall, normalmode, totalwords,

         totalcharacters;

   JScrollPane jsp;

   FileDialog fd;

   JDialog words;

   JButton Bold, italic, plain, fsizeok, panelclose;

   JPanel Taskbar, bottom ,center;

   JTextField fSize,Title;


   // constructor of the main class

   Anotepad() {

      super("Advance NotePad");


      // creating objects of all components

      ta = new JTextArea("//Here you can start your writing ");

      jsp = new JScrollPane(ta);

      fd = new FileDialog(new Frame());
```

```
menubar = new JMenuBar();

center = new JPanel();

center.setLayout(new BorderLayout());

Title = new JTextField("Enter Title here",100);

menu1 = new JMenu("File");

item1 = new JMenuItem("Write Notes");

item2 = new JMenuItem("exit");

item3 = new JMenuItem("clear");

item5 = new JMenuItem("save");

item6 = new JMenuItem("Notepad");

open = new JMenuItem("open");

fontsize = new JMenuItem("Font");

menu2 = new JMenu("Edit");

item4 = new JMenuItem("Dark Mode");

cut = new JMenuItem("Cut");

copy = new JMenuItem("Copy");

paste = new JMenuItem("Paste");

selectall = new JMenuItem("Select All");

menu3 = new JMenu("view");

normalmode = new JMenuItem("Normal Mode");

totalwords = new JMenuItem("Total Words");

totalcharacters = new JMenuItem("Total Characters");

Bold = new JButton("Bold");

Bold.setFont(new Font("Courier", Font.BOLD, 10));

italic = new JButton("Italic");

italic.setFont(new Font("Courier", Font.ITALIC, 10));

plain = new JButton("Plain");

plain.setFont(new Font("Courier", Font.PLAIN, 10));

panelclose = new JButton("X");
```

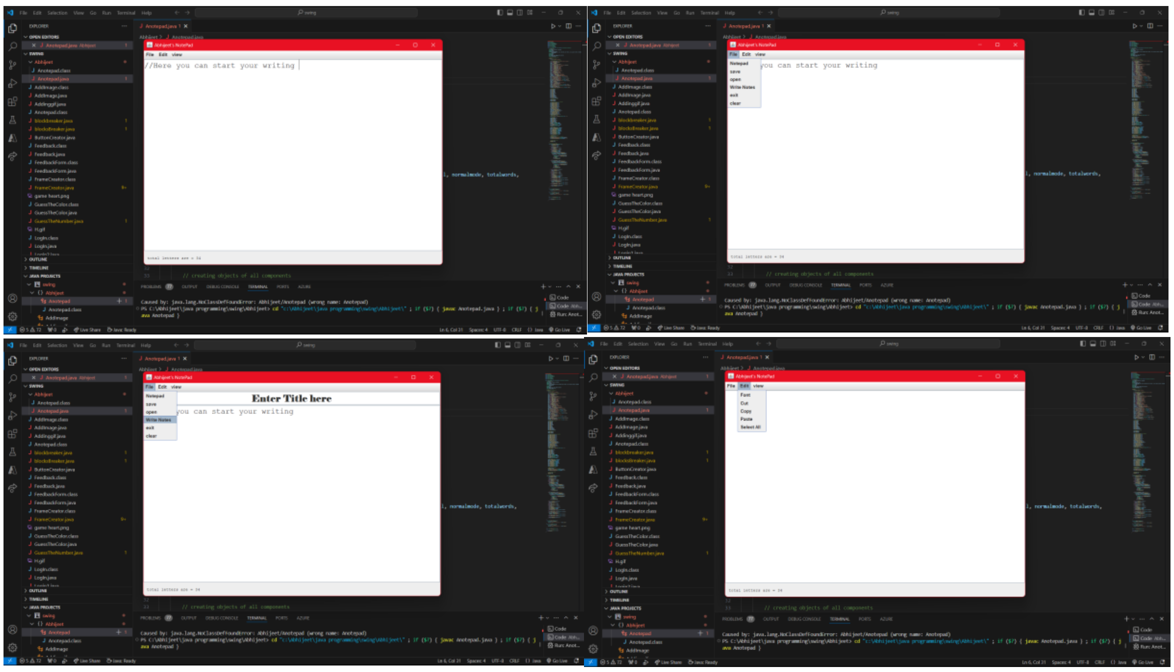
```
panelclose.setFont(new Font("Courier", Font.BOLD, 10));
Taskbar = new JPanel();
Taskbar.setVisible(false);
labelfsize = new JLabel("Enter Font Size: ");
fSize = new JTextField(3);
fsizeok = new JButton("OK");
fsizeok.setFont(new Font("Courier", Font.PLAIN, 10));
bottom = new JPanel();
bottom.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 10));
bottom.setVisible(true);
textLength = new JLabel("");
textLength.setFont(new Font("Courier", Font.PLAIN, 10));

// adding menu items to menu
menu1.add(item6);
menu1.add(item5);
menu1.add(open);
menu1.add(item1);
menu1.add(item2);
menu1.add(item3);
menu2.add(fontsize);
menu2.add(cut);
menu2.add(copy);
menu2.add(paste);
menu2.add(selectall);
menu3.add(item4);
menu3.add(normalmode);
menu3.add(totalwords);
menu3.add(totalcharacters);
```

```
// adding menu to menu bar
menubar.add(menu1);
menubar.add(menu2);
menubar.add(menu3);

// adding actionListener to all components
item1.addActionListener(this);
item2.addActionListener(this);
item3.addActionListener(this);
item4.addActionListener(this);
item5.addActionListener(this);
item6.addActionListener(this);
open.addActionListener(this);
fontsize.addActionListener(this);
cut.addActionListener(this);
copy.addActionListener(this);
paste.addActionListener(this);
selectall.addActionListener(this);
normalmode.addActionListener(this);
totalcharacters.addActionListener(this);
totalwords.addActionListener(this);
Bold.addActionListener(this);
italic.addActionListener(this);
fsizeok.addActionListener(this);
plain.addActionListener(this);
panelclose.addActionListener(this);
ta.addKeyListener(this);
```

```
// adding layout to frame
Taskbar.setLayout(new FlowLayout(FlowLayout.LEFT, 2, 2));

// add Components
Taskbar.add(panelclose);
Taskbar.add(Bold);
Taskbar.add(italic);
Taskbar.add(plain);
Taskbar.add(labelfsize);
Taskbar.add(fSize);
Taskbar.add(fsizeok);
bottom.add(textLength);
add(Taskbar, BorderLayout.NORTH);
add(center, BorderLayout.CENTER);
center.add(Title,BorderLayout.NORTH);
Title.setVisible(false);
center.add(jsp, BorderLayout.CENTER);
add(bottom, BorderLayout.SOUTH);

// adding colors
Taskbar.setBackground(Color.LIGHT_GRAY);
panelclose.setBackground(Color.RED);
panelclose.setForeground(Color.WHITE);
Bold.setBackground(Color.DARK_GRAY);
italic.setBackground(Color.DARK_GRAY);
plain.setBackground(Color.DARK_GRAY);
fsizeok.setBackground(Color.DARK_GRAY);
Bold.setForeground(Color.WHITE);
italic.setForeground(Color.WHITE);
```

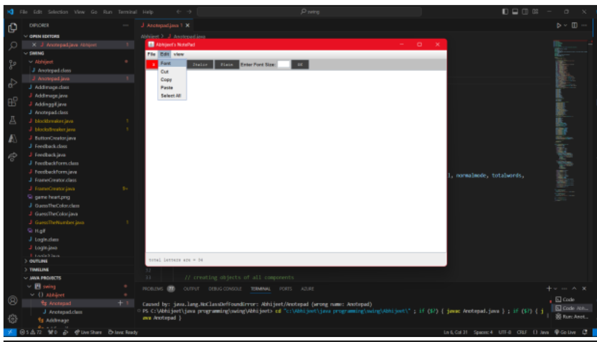
```
        plain.setForeground(Color.WHITE);
        fsizeok.setForeground(Color.WHITE);
        Title.setBackground(Color.white);

     // adding menu bar to frame
        setJMenuBar(menubar);

   // giving font and size to text of textArea
        ta.setFont(new Font("Courier", Font.PLAIN, fontsize1));
        Title.setFont(new Font("Elephant",Font.BOLD,fontsize1+5));
        Title.setHorizontalAlignment(JTextField.CENTER);

     // basic methods of frame
        setSize(800, 600);
        setVisible(true);
      }
      // giving action to buttons
      public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == item6) {
          Title.setVisible(false);
        }
        if (ae.getSource() == item1) {
          Title.setVisible(true);
        } else if (ae.getSource() == item2) {
          this.dispose();
        } else if (ae.getSource() == item3) {
          ta.setText("");
        } else if (ae.getSource() == item4) {
          ta.setBackground(Color.BLACK);
        ta.setForeground(Color.WHITE);
    } else if (ae.getSource() == item5) {
        JFileChooser fileChooser = new JFileChooser();
        int returnValue = fileChooser.showSaveDialog(this);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File fileToSave = fileChooser.getSelectedFile();
            try {
                FileWriter fw = new FileWriter(fileToSave);
                BufferedWriter bw = new BufferedWriter(fw);
                ta.write(bw);
                bw.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    } else if (ae.getSource() == open) {
        JFileChooser fileChooser = new JFileChooser();
        int returnValue = fileChooser.showOpenDialog(this);
        if (returnValue == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            try {
                FileReader fr = new FileReader(selectedFile);
                BufferedReader br = new BufferedReader(fr);
                ta.read(br, null);
                br.close();
                ta.requestFocus();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    } else if (ae.getSource() == fontsize) {
        Taskbar.setVisible(true);
    } else if (ae.getSource() == cut) {
        ta.cut();
    } else if (ae.getSource() == copy) {
        ta.copy();
    } else if (ae.getSource() == paste) {
        ta.paste();
    } else if (ae.getSource() == selectall) {
        ta.selectAll();
    } else if (ae.getSource() == normalmode) {
        ta.setBackground(Color.WHITE);
        ta.setForeground(Color.BLACK);
    } else if (ae.getSource() == Bold) {
        ta.setFont(new Font("Courier", Font.BOLD, fontsize1));
    } else if (ae.getSource() == italic) {
        ta.setFont(new Font("Courier", Font.ITALIC, fontsize1));
    } else if (ae.getSource() == fsizeok) {
        String s1 = fSize.getText();
        fontsize1 = Integer.parseInt(s1);
        ta.setFont(new Font("Courier", Font.PLAIN, fontsize1));
    } else if (ae.getSource() == plain) {
        ta.setFont(new Font("Courier", Font.PLAIN, fontsize1));
    } else if (ae.getSource() == panelclose) {
        Taskbar.setVisible(false);
    }
}
public void keyPressed(KeyEvent ke) {
        String letters = ta.getText();
        textLength.setText("total letters are = " + letters.length());
    }
    public void keyTyped(KeyEvent ke) {
        String letters = ta.getText();
        textLength.setText("total letters are = " + letters.length());
    }
    public void keyReleased(KeyEvent ke) {
    }
    public static void main(String[] args) {
        new Anotepad();
    }
}
```

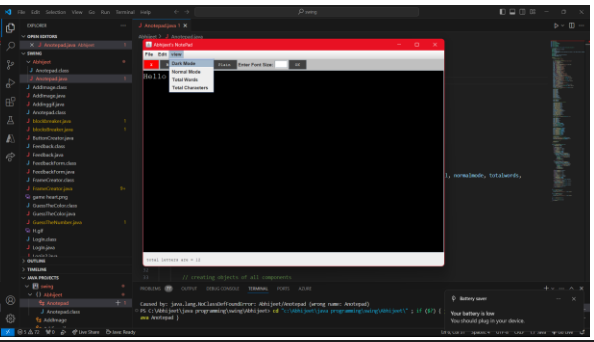
Output :

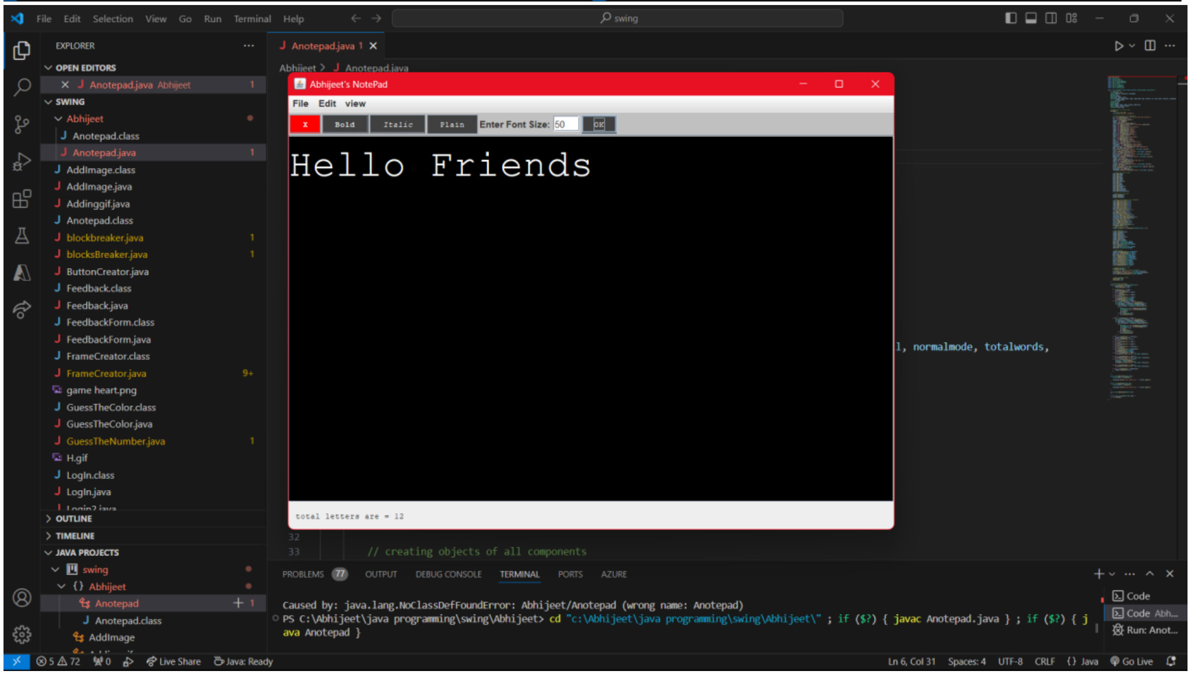


```
//Here you can start your writing
total letters are = 34
```

```
Caused by: java.lang.NoClassDefFoundError: Abhijeet/Anotepad (wrong name: Anotepad)
PS C:\Abhijeet\java programming\swing\Abhijeet> cd "c:\Abhijeet\java programming\swing\Abhijeet\" ; if ($?) { javac Anotepad.java } ; if ($?) { java Anotepad }
```







```
Hello Friends

total letters are = 12
```

```
// creating objects of all components
```

```
Caused by: java.lang.NoClassDefFoundError: Abhijeet/Anotepad (wrong name: Anotepad)
PS C:\Abhijeet\java programming\swing\Abhijeet> cd "c:\Abhijeet\java programming\swing\Abhijeet\" ; if ($?) { javac Anotepad.java } ; if ($?) { j
ava Anotepad }
```

9. Conclusion

The Advanced Notepad project delivers a robust and feature-rich text editing application built using Java programming language and Swing GUI library. The application provides users with a convenient platform for creating, editing, and managing text documents while offering various advanced features for enhanced productivity. Future enhancements can further improve the functionality and usability of the application, making it a valuable tool for users.

10. References

Oracle Documentation: https://docs.oracle.com/javase/8/docs/

Java Swing Tutorial: https://docs.oracle.com/javase/tutorial/uiswing/