A
PROJECT REPORT ON

"NOTEPAD APP"

By

Sr. No.	NAME	ROLL NO.
1	Vedant Narawadkar	32151
2	Aarya Sonawane	32171

GUIDE
MR. NILESH S. SHIRUDE

DEPARTMENT OF
ELECTRONICS AND TELECOMMUNICATION ENGINEERING
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
PUNE – 43

A.Y. 2024-25

INDEX

1. PROBLEM STATEMENT:

Develop a simple text-editing application that allows users to create, edit, and manage plain text documents efficiently. The application should provide essential functionalities such as creating new documents, opening existing text files, saving changes, and printing content. Additionally, it should support basic editing features like copy, paste, cut, and select all, accessible through a user-friendly menu interface. The system must include a help option to display information about the application. The goal is to offer a lightweight, intuitive notepad tool that enhances user productivity for basic text manipulation tasks.

2. OBJECTIVE :

- **Implement Text Editing Functionality:** Develop a system that enables users to create, edit, and save plain text documents with ease, using a central text area for input and modification.
- **Provide File Management Features:** Accurately handle file operations such as creating new documents, opening existing text files, saving content, and printing documents.
- **Display Real-Time Text Updates:** Continuously display the user's text input and edits in real-time within the application's text area.
- **Incorporate Keyboard Shortcuts:** Integrate intuitive keyboard shortcuts (e.g., Ctrl+N, Ctrl+S) to streamline access to core functionalities like new, save, and open.
- **Ensure Cross-Platform Compatibility:** Design the application to be responsive and functional across various screen sizes and operating systems using Java Swing.
- **Implement Basic Editing Tools:** Enable essential editing features such as copy, paste, cut, and select all to enhance text manipulation efficiency.
- **Design Intuitive User Interface:** Create a clean and user-friendly menu-based interface that clearly organizes file, edit, and help options without overwhelming the user.
- **Provide Application Information:** Include an "About" section under the help menu to display relevant details about the notepad application.
- **Optimize Performance:** Ensure the application runs smoothly without delays, particularly during file operations and text editing tasks.

- **Enhance User Productivity:** Design the notepad to be simple yet effective, encouraging users to efficiently manage and edit text for everyday tasks.

3. INTRODUCTION:

3.1 Background/context

The notepad application project focuses on developing a lightweight, efficient, and user-friendly tool for creating and managing plain text documents. Text editors like this have been fundamental to computing since its early days, serving as essential utilities for programmers, writers, and everyday users. The concept of a simple text editor traces back to the 1960s and 1970s, with tools like ed and later Notepad on Windows, which provided a no-frills platform for text manipulation. Over time, these tools have evolved to meet the growing demands of users, balancing simplicity with functionality, and remain relevant in a world of complex software suites.

With advancements in programming languages and graphical user interfaces, modern text editors have become more accessible, offering intuitive designs and practical features for a wide range of applications—from coding and note-taking to drafting documents. This project builds on that tradition by leveraging Java Swing to create a minimalistic yet effective notepad application. It aims to enhance user productivity through essential file management and editing capabilities, while maintaining a straightforward interface that appeals to both novice and experienced users. The application serves as a practical tool for everyday text-based tasks, reflecting the enduring utility of simple text editors in today's digital landscape.

3.2 Relevance

This project is relevant to Computer Science and Software Engineering as it emphasizes the development of a functional, user-centric application with real-time text processing and file management capabilities. The notepad application involves handling user inputs (such as text edits and menu selections) and performing file operations efficiently, which ties into core concepts like event-driven programming, graphical user interface (GUI)

design, and system resource management. While primarily software-focused, it connects to broader topics such as human-computer interaction and application optimization, which are critical in designing practical tools and systems. This project provides hands-on experience in building a responsive, lightweight application using Java Swing, fostering skills in software development, user experience design, and system performance—abilities that are highly applicable in creating utility software and other interactive applications.

3.3 Project Details

The Notepad Application is a straightforward Java Swing-based text editor designed to provide users with a simple yet effective tool for creating, editing, and managing plain text documents. Below are the key details and features:

- The application features a central text area where users can input and modify text.
- It includes a menu bar with options for file operations, editing functions, and help.
- Users can create new documents, open existing text files, save changes, and print content.
- Basic editing tools like copy, paste, cut, and select all are supported with keyboard shortcuts.
- An "About" window provides information about the application, including its purpose and version.

Key Features:

1. **UI & Graphics:**

- Utilizes Swing (JFrame, JTextArea, JMenuBar) for a clean and responsive interface.
- Displays a notepad icon and Windows logo in the "About" window for visual appeal.
- Text is rendered in a customizable font (San Serif, size 20) with word wrap enabled.

2. **File and Editing Logic:**

- **File Operations:** Supports "New" (Ctrl+N), "Open" (Ctrl+O), "Save" (Ctrl+S), and "Print" (Ctrl+P) via a JFileChooser for file handling.
- **Editing Tools:** Implements "Copy" (Ctrl+C), "Paste" (Ctrl+V), "Cut" (Ctrl+X), and "Select All" (Ctrl+A) for efficient text manipulation.

3. **Help & Accessibility:**

- An "About Notepad" option under the Help menu opens a separate window with application details.
- The application exits gracefully using "Exit" (Esc) or the window close button.

3.4 Scope**:**

The Notepad Application is a small-scale text-editing tool developed using Java Swing, designed to provide essential functionality for managing plain text documents.

User Interaction:

- Users can create, edit, and save text documents within a central text area.
- The application supports file operations like opening and saving .txt files, as well as printing content.
- Basic editing features (copy, paste, cut, select all) are accessible via menu options or keyboard shortcuts.
- An "About" window provides application details, and the program can be exited cleanly via the menu or window controls.

Technologies Used:

- **Java Swing:** For building the graphical user interface (JFrame, JTextArea, JMenuBar, etc.).
- **AWT (Abstract Window Toolkit):** For event handling and basic layout management.
- **Java I/O & ActionListener:** For file operations (FileReader, BufferedWriter) and responding to user actions (menu clicks, shortcuts).

4.SOURCE CODE:

Paste working source code and illustrate the same

Notepad.java:

```
package notepad;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import javax.swing.filechooser.*;

public class Notepad extends JFrame implements ActionListener {

	private JTextArea area;
	private JScrollPane scpane;
	String text = "";

	public Notepad() {
		super("Notepad");
		setSize(1950, 1050);

		setLayout(new BorderLayout());

		JMenuBar menuBar = new JMenuBar(); // menubar

		JMenu file = new JMenu("File"); // file menu

		JMenuItem newdoc = new JMenuItem("New");
		newdoc.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,
ActionEvent.CTRL_MASK));
		newdoc.addActionListener(this);

		JMenuItem open = new JMenuItem("Open");
		open.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
ActionEvent.CTRL_MASK));
		open.addActionListener(this);

		JMenuItem save = new JMenuItem("Save");
		save.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
ActionEvent.CTRL_MASK));
		save.addActionListener(this);

		JMenuItem print = new JMenuItem("Print");
		print.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P,
ActionEvent.CTRL_MASK));
```

```
            print.addActionListener(this);

            JMenuItem exit = new JMenuItem("Exit");
            exit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0));
            exit.addActionListener(this);

            JMenu edit = new JMenu("Edit");

            JMenuItem copy = new JMenuItem("Copy");
            copy.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,
ActionEvent.CTRL_MASK));
            copy.addActionListener(this);

            JMenuItem paste = new JMenuItem("Paste");
            paste.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V,
ActionEvent.CTRL_MASK));
            paste.addActionListener(this);

            JMenuItem cut = new JMenuItem("Cut");
            cut.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,
ActionEvent.CTRL_MASK));
            cut.addActionListener(this);

            JMenuItem selectall = new JMenuItem("Select All");
            selectall.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A,
ActionEvent.CTRL_MASK));
            selectall.addActionListener(this);

            JMenu about = new JMenu("Help");

            JMenuItem notepad = new JMenuItem("About Notepad");
            notepad.addActionListener(this);

            area = new JTextArea();
            area.setFont(new Font("SAN_SERIF", Font.PLAIN, 20));
            area.setLineWrap(true);
            area.setWrapStyleWord(true);

            scpane = new JScrollPane(area);
            scpane.setBorder(BorderFactory.createEmptyBorder());

            setJMenuBar(menuBar);
            menuBar.add(file);
            menuBar.add(edit);
            menuBar.add(about);

            file.add(newdoc);
            file.add(open);
            file.add(save);
            file.add(print);
            file.add(exit);

            edit.add(copy);
            edit.add(paste);
            edit.add(cut);
```

```
		edit.add(selectall);

		about.add(notepad);

		add(scpane, BorderLayout.CENTER);
		setVisible(true);
	}

	public void actionPerformed(ActionEvent ae) {
		if (ae.getActionCommand().equals("New")) {
			area.setText("");

		} else if (ae.getActionCommand().equals("Open")) {
			JFileChooser chooser = new JFileChooser("D:/Java");
			chooser.setAcceptAllFileFilterUsed(false);
			FileNameExtensionFilter restrict = new
FileNameExtensionFilter("Only .txt files", "txt");
			chooser.addChoosableFileFilter(restrict);

			int result = chooser.showOpenDialog(this);
			if (result == JFileChooser.APPROVE_OPTION) {
				File file = chooser.getSelectedFile();

				try {
					System.out.println("HEki");
					FileReader reader = new FileReader(file);
					BufferedReader br = new BufferedReader(reader);
					area.read(br, null);
					br.close();
					area.requestFocus();
				} catch (Exception e) {
					System.out.print(e);
				}
			}
		} else if (ae.getActionCommand().equals("Save")) {
			final JFileChooser SaveAs = new JFileChooser();
			SaveAs.setApproveButtonText("Save");
			int actionDialog = SaveAs.showOpenDialog(this);
			if (actionDialog != JFileChooser.APPROVE_OPTION) {
				return;
			}

			File fileName = new File(SaveAs.getSelectedFile() + ".txt");
			BufferedWriter outFile = null;
			try {
				outFile = new BufferedWriter(new FileWriter(fileName));
				area.write(outFile);
			} catch (IOException e) {
				e.printStackTrace();
			}
		} else if (ae.getActionCommand().equals("Print")) {
			try {
				area.print();
			} catch (Exception e) {
			}
```

```
            } else if (ae.getActionCommand().equals("Exit")) {
                  System.exit(0);
            } else if (ae.getActionCommand().equals("Copy")) {
                  text = area.getSelectedText();
            } else if (ae.getActionCommand().equals("Paste")) {
                  area.insert(text, area.getCaretPosition());
            } else if (ae.getActionCommand().equals("Cut")) {
                  text = area.getSelectedText();
                  area.replaceRange("", area.getSelectionStart(),
area.getSelectionEnd());
            } else if (ae.getActionCommand().equals("Select All")) {
                  area.selectAll();
            } else if (ae.getActionCommand().equals("About Notepad")) {
                  new About().setVisible(true);

            }
      }

      public static void main(String[] args) {
            new Notepad();
      }
}
```

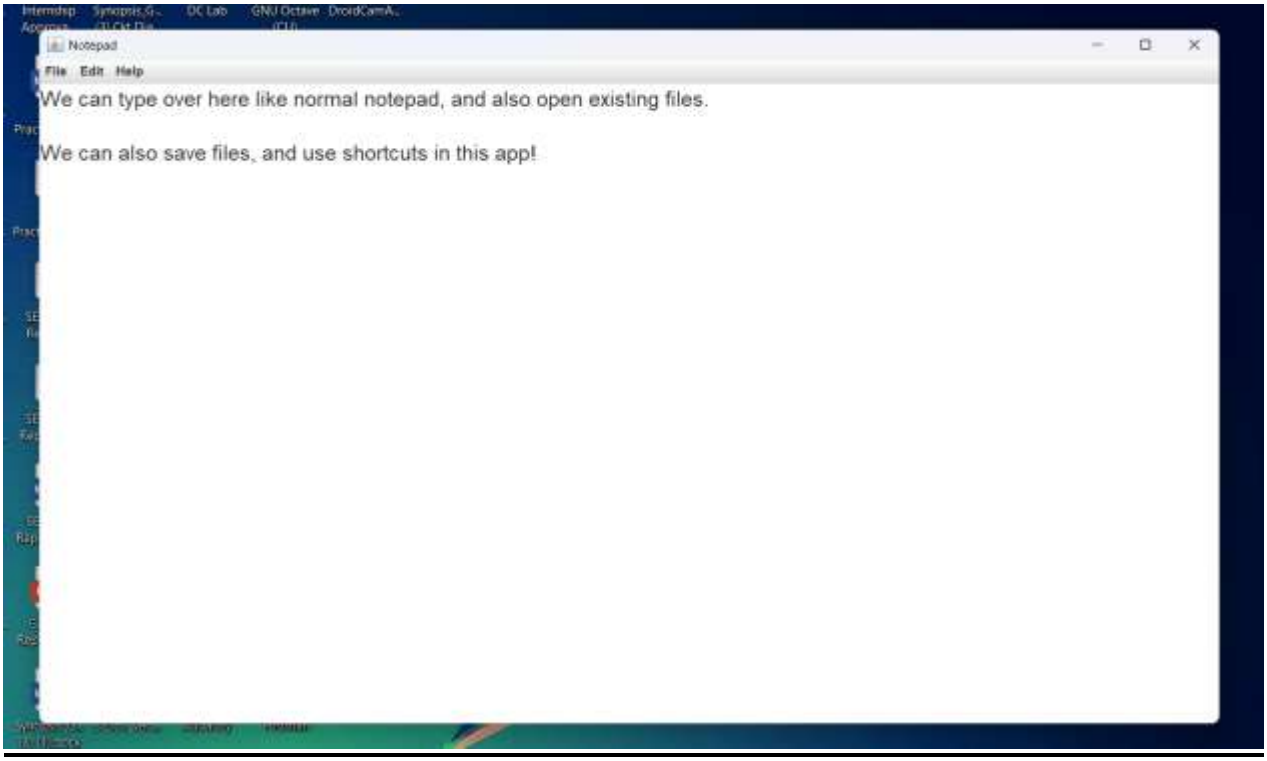
About.java:

```
package notepad;

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

public class About extends JFrame implements ActionListener {
      JButton b1;

      About() {
            setBounds(600, 200, 700, 600);
            setLayout(null);

            ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("notepad/icons/windows.png"));
            Image i2 = i1.getImage().getScaledInstance(400, 80,
Image.SCALE_DEFAULT);
            ImageIcon i3 = new ImageIcon(i2);
            JLabel l1 = new JLabel(i3);
            l1.setBounds(150, 40, 400, 80);
            add(l1);

            ImageIcon i4 = new
ImageIcon(ClassLoader.getSystemResource("notepad/icons/notepad.png"));
            Image i5 = i4.getImage().getScaledInstance(70, 70, Image.SCALE_DEFAULT);
            ImageIcon i6 = new ImageIcon(i5);
            JLabel l2 = new JLabel(i6);
            l2.setBounds(50, 180, 70, 70);
```

```
            add(l2);

            JLabel l3 = new JLabel(
                        "<html>Advanced Java Programming<br>Vedant
Narawadkar<br>All rights reserved<br><br>Notepad is a word processing program,
<br>which allows changing of text in a computer file.<br>Notepad is a simple text
editor for basic text-editing program<br> which enables computer users to create
documents. </html>");
            l3.setFont(new Font("SAN_SERIF", Font.PLAIN, 18));
            l3.setBounds(150, 130, 500, 300);
            add(l3);

            b1 = new JButton("OK");
            b1.setBounds(580, 500, 80, 25);
            b1.addActionListener(this);
            add(b1);

      }

      public void actionPerformed(ActionEvent ae) {
            this.setVisible(false);
      }

      public static void main(String[] args) {
            new About().setVisible(true);
      }
}
```

5.RESULT

Screen shot of output



6. CONCLUSION:

The Notepad Application successfully demonstrates the implementation of a simple yet functional text-editing tool using Java Swing. The application effectively enables users to create, edit, and manage plain text documents with ease, incorporating essential features like file operations and basic editing tools. The intuitive menu-based interface, combined with keyboard shortcuts, enhances user productivity and accessibility. The project highlights efficient event handling, graphical user interface design, and file management capabilities, making it a practical exercise in software development and user experience design. This notepad serves as a solid foundation for future enhancements, such as advanced text formatting, cloud integration, or support for additional file types.

7.APPLICATIONS:

- **Skill Development & Training:** Enhances text-editing efficiency, familiarity with keyboard shortcuts, and basic software navigation, making it useful for beginner-level computer literacy training.

- **Software Development Industry:** Provides a foundational example of a lightweight utility application, which can be expanded into more complex text editors or integrated into larger software suites.
- **User Interaction Research:** Supports studies in human-computer interaction, usability testing, and GUI design by offering a simple platform for analyzing user behavior with text-based tools.
- **Productivity Tools:** Can be adapted for quick note-taking, drafting, or editing tasks, aiding professionals, students, and casual users in managing text efficiently.
- **Education & Learning Tools:** Serves as an engaging way to introduce programming concepts like event-driven programming, file I/O operations, and Swing-based UI development to students.
- **Programming Practice:** Offers a practical project for developers to hone skills in Java, object-oriented design, and application optimization for real-world use cases.
- **Everyday Utility & Casual Use:** Provides a simple, reliable text-editing solution for personal use, suitable for users of all skill levels seeking a no-frills notepad experience.

8. FUTURE SCOPE:

- **Multi-Document Support:** The application can be enhanced to support multiple open documents simultaneously, allowing users to switch between tabs or windows for improved multitasking.
- **Mobile Version:** The notepad could be adapted into a mobile app for Android and iOS, broadening its accessibility and leveraging touch-based text input.
- **Advanced Text Formatting:** Features like font selection, text size adjustment, or basic formatting (bold, italic) could be added to increase functionality while maintaining simplicity.
- **Cloud Integration:** The application could incorporate cloud storage options (e.g., Google Drive, Dropbox) to enable seamless saving, syncing, and accessing of files across devices.
- **Customization Options:** Users could customize the interface, such as themes, font styles, or window layouts, to enhance the visual and functional experience.
- **Search and Replace Functionality:** Adding a search-and-replace feature would improve text editing efficiency, making it more competitive.

9. COPY RIGHT AFFIRMATION:

We undersigned pledge and represent that the source code printed in this mini project report does not violate any proprietary or personal rights of others (including, without limitation, any copyrights or privacy rights); that the Work is factually accurate and contains no matter libellous or otherwise unlawful; that we have substantially participated in the creation of the Work and that it represents our original work sufficient for us to claim authorship.

Name of students **Sign**

1. **Vedant Narawadkar**
2. **Aarya Sonawane**