

**Shripriti**

Educational & IT Hub

Step towards your Success

**Project Report**

**On**

**Notepad Clone**

**Submitted by**

**Chetan Kailas Banait**

**MCA II YEAR**

**SEM IV**

**Under the Guidance of**

**Prof. Pranjali Ulhe**

**Assistant Professor**

**Submitted to**

**Shripriti**

**Educational & IT Hub**

**Academic Year 2025-26**



“**Notepad Clone**”



**Shripriti**

Educational & IT Hub

Step towards your Success

## DECLARATION

We hereby declare that the Project work titled “Notepad Clone” submitted to Shripriti Educational & IT Hub is a record of an original work done by us under the guidance of Prof. Pranjali Ulhe, Assistant Professor, School of Allied Sciences, Datta Meghe Institute of Higher Education & Research, Sawangi (Meghe) Wardha in the AY 2025-26 for a partial fulfillment of requirement.

This report has not been submitted to any other University or Institute for the award of any Internship

**MCA II Year Semester – IV : Chetan Kailas Banait**

**Place: Wardha, MH**

**Date: \_\_/\_\_/\_\_\_\_**



**Shripriti**

Educational & IT Hub

Step towards your Success

## ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude’s to our Project guide Prof. Pranjali Ulhe, Faculty of Science and Technology, School of Allied Sciences, Datta Meghe Institute of Higher Education & Research, Sawangi (Meghe), Wardha for his/her able guidance and support in completing this report. We would like to extend my gratitude to and **Shripriti Educational & IT Hub ,** for providing me with all the facility that was required to complete this report successfully.

We also thank the management of **Shripriti Educational & IT Hub**  for providing me/us state of the art infrastructure and the opportunity to embark this Project. At last but not the least we are thankful to all my all teachers, staff who have been always helping and encouraging me/us throughout the period of this project.

**MCA II Year Semester – IV : Chetan Kailas Banait**

**Place: Wardha, MH**

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Chapter Name** | **Page No** |
| 1 | Cover Page |  |
| 2 | Acknowledgment |  |
| 3 | Table of Contents |  |
| 4 | Introduction • Purpose of the Internship • Overview of Expectations • Technologies and Stack Covered |  |
| 5 | Weekly Breakdown • Week-by-week Summary • Challenges and Solutions |  |
| 6 | Mini Projects • Description and Features • Screenshots |  |
| 7 | Final Project:  • Project Overview  • Objective and Motivation  • Features Implemented  • Frontend & Backend Technologies Used  • Database Schema  • APIs Created  • Deployment Details |  |
| 8 | Learning Outcomes • Key Takeaways • Tools Learned • Real-world Experience |  |
| 9 | Conclusion • Internship Impact • Future Scope |  |
| 10 | Appendix • Code Snippets • GitHub Links • Extra Screenshots • References |  |

### Introduction

## ****Purpose of the Internship****

The purpose of this internship was to gain practical experience in Java-based desktop application development by creating a functional **Notepad Clone Project**. This project helped solidify my understanding of Java GUI programming, file handling, and user interaction through intuitive interfaces. By building a simplified version of a classic text editor, I aimed to bridge the gap between academic knowledge and real-world application development. It also served to enhance my logical thinking and software development discipline.

### ****Overview of Expectations****

During the internship, I was expected to:

* Understand the functional requirements of a basic text editor.
* Design and implement a user-friendly interface using Java Swing.
* Implement features such as open, save, copy, paste, cut, find, replace, and font customization.
* Learn and apply file handling in Java for managing text file input and output operations.
* Test the application thoroughly for usability and bugs.
* Follow good coding practices and provide inline comments and documentation.
* Optionally, implement additional features like dark mode, tab support, or word count.

## ****Technologies and Stack Covered****

### ****Technologies Used:****

**Programming Language:** Java

**GUI Libraries:**

**AWT (Abstract Window Toolkit):** Utilized for basic GUI components such as Labels and MenuItems.

**Swing:** Used to design the complete UI, including components like JFrame, JTextArea, JMenuBar, JMenu, JFileChooser, and more.

**Database:** Not required for this project (File handling is the primary mode of storage).

**IDE Used:** IntelliJ IDEA / NetBeans / Eclipse (depending on user preference)

### ****Key Concepts Practiced:****

**Event Handling:** Used for menu actions like open, save, copy, paste, etc.

**Exception Handling:** Implemented to manage file-related errors and user input issues.

**File Handling:** Read and write operations on text files using BufferedReader, BufferedWriter, FileReader, and FileWriter.

**Swing Layout Management:** Efficiently used layout managers like BorderLayout, FlowLayout, etc.

**Modularity and Reusability:** Functions and listeners were structured modularly for better readability and reusability.

**Weekly Breakdown**

### ****Week 1: Project Overview & Setup****

**Summary:**  
The first week focused on understanding the scope of the Notepad Clone Project and setting up the development environment. I explored the basic structure of a text editor application and planned out the key features like open, save, edit, and formatting.

**Challenges & Solutions:**  
Initial challenge was setting up the GUI window and integrating menu bars using Swing. This was solved by referring to Java Swing documentation and building a basic JFrame with JMenuBar.

### ****Week 2: Core Functionalities – File Operations****

**Summary:**  
Implemented the fundamental file operations: creating new files, opening existing ones, and saving files using JFileChooser. Ensured that file contents were properly loaded into and saved from the JTextArea.

**Challenges & Solutions:**  
The main challenge was ensuring correct file encoding and preventing data loss. I resolved this by using BufferedReader and BufferedWriter with proper exception handling and user confirmation prompts.

### ****Week 3: Edit Functionalities – Cut, Copy, Paste, Undo****

**Summary:**  
Developed core editing functionalities such as cut, copy, paste, and undo. Integrated these with the JTextArea using inbuilt methods and listeners.

**Challenges & Solutions:**  
The undo feature was tricky. I solved this by integrating UndoManager from javax.swing.undo package, which allowed tracking and reversing text changes effectively.

### ****Week 4: Formatting Features – Font and Color Options****

**Summary:**  
Added features to change font styles, sizes, and text colors. Implemented font selection dialogs and integrated JColorChooser for color customization.

**Challenges & Solutions:**  
Managing consistent styling across the entire text was challenging. I used setFont() and setForeground() functions carefully to maintain formatting integrity.

### ****Week 5: Advanced Features – Find & Replace, Word Count****

**Summary:**  
Implemented search functionality using dialog boxes and highlighted search results. Added a word and character counter displayed in the status bar.

**Challenges & Solutions:**  
Implementing real-time word count updates was tricky. I solved this by adding a DocumentListener to monitor changes in the JTextArea.

### ****Week 6: File Export and Printing Support****

**Summary:**  
Added the ability to print the contents of the notepad and export them as a .txt file. Integrated Java’s PrinterJob class to provide printing support.

**Challenges & Solutions:**  
Printing layout setup was a challenge. I overcame it by customizing the Printable interface and properly setting margins and headers.

### ****Week 7: Final Testing & Bug Fixing****

**Summary:**  
Performed thorough testing of all features including edge cases like empty files, large files, or user cancellations. Ensured stability across all functions.

**Challenges & Solutions:**  
A few bugs in file overwrite warnings and undo limits were found. Fixed by enhancing dialog confirmations and managing undo queue size.

### ****Week 8: Documentation & Final Presentation****

**Summary:**  
Documented the entire project covering features, usage instructions, and code structure. Prepared a user manual and delivered a final presentation of the project to mentors.

**Challenges & Solutions:**  
Writing comprehensive documentation and organizing code comments was time-consuming. I tackled this by documenting each class and method individually, supported with screenshots and flowcharts.

#### Mini Projects

### ****1. Notepad Clone****

**Description:**  
This mini project is a **desktop-based text editor application** developed using **Core Java, AWT, and Swing**. The Notepad Clone replicates the functionality of the basic Windows Notepad, offering a simple and clean interface for writing, editing, and managing text files. It is an ideal tool for learning GUI development and file handling in Java.

**Key Features:**

**Create New File:**  
Start a new text document within the application.

**Open Existing File:**  
Load and view the contents of existing .txt files using Java’s File Dialog and FileReader.

**Save File:**  
Save the written content to a file using FileWriter, supporting both save and save-as options.

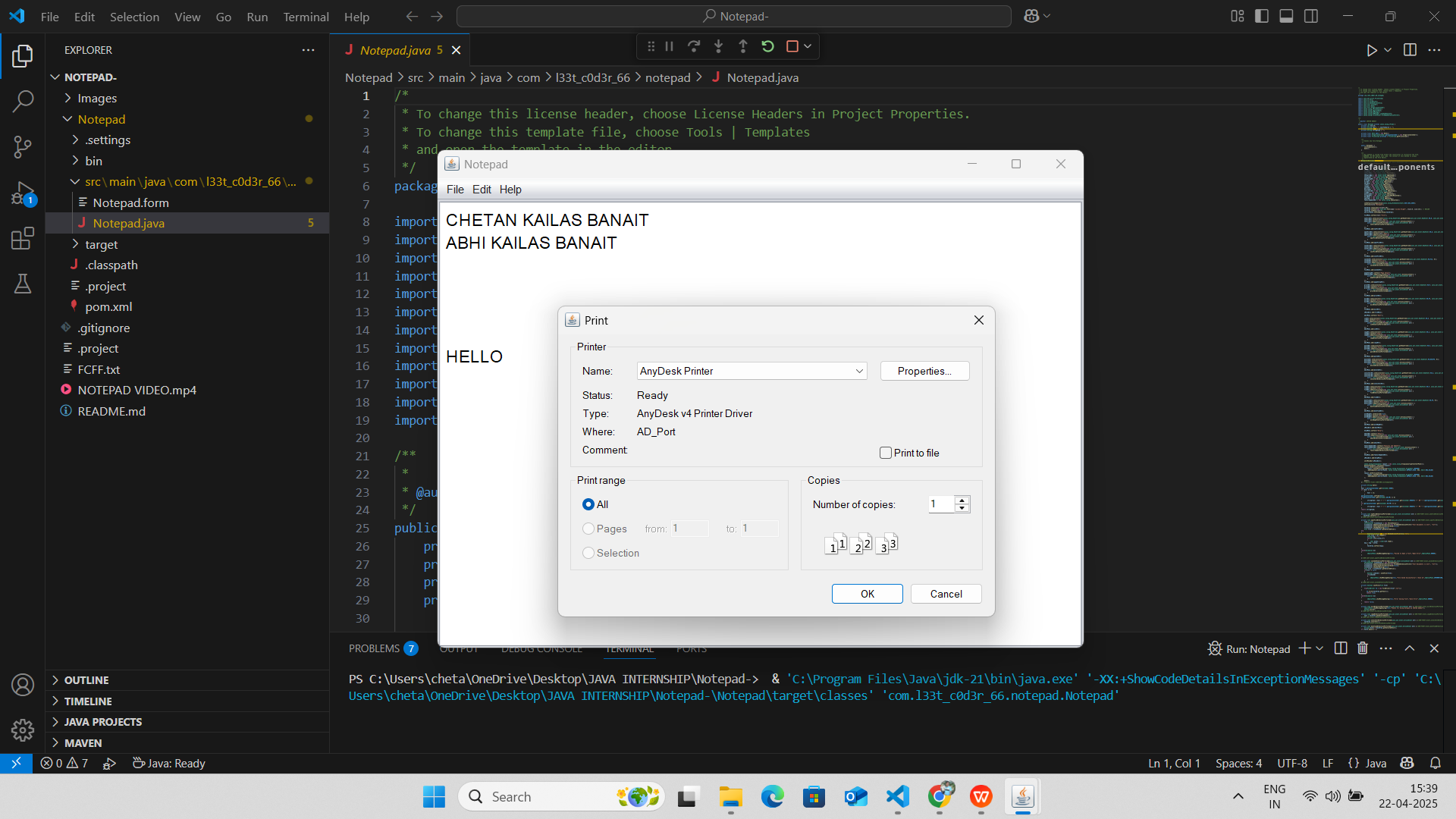
**Edit Functions:**  
Includes basic editing features like cut, copy, paste, select all, and undo/redo functionality.

**Text Formatting:**  
Allows changing font style, size, and color to enhance readability and customization.

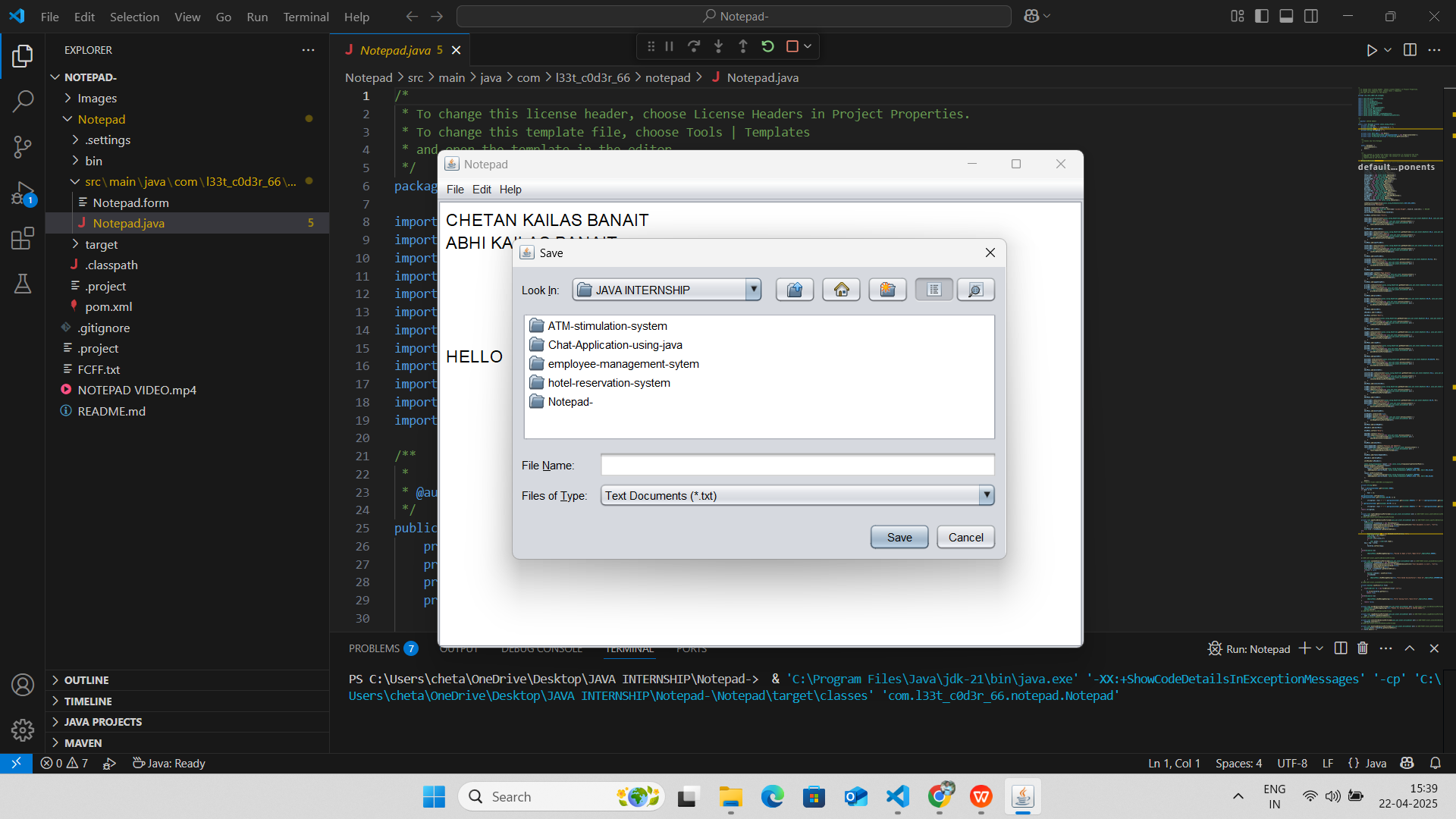
**Find and Replace:**  
Enables users to search for specific words or phrases and optionally replace them with new text.

**Status Bar (optional):**  
Displays cursor position, word count, or character count.

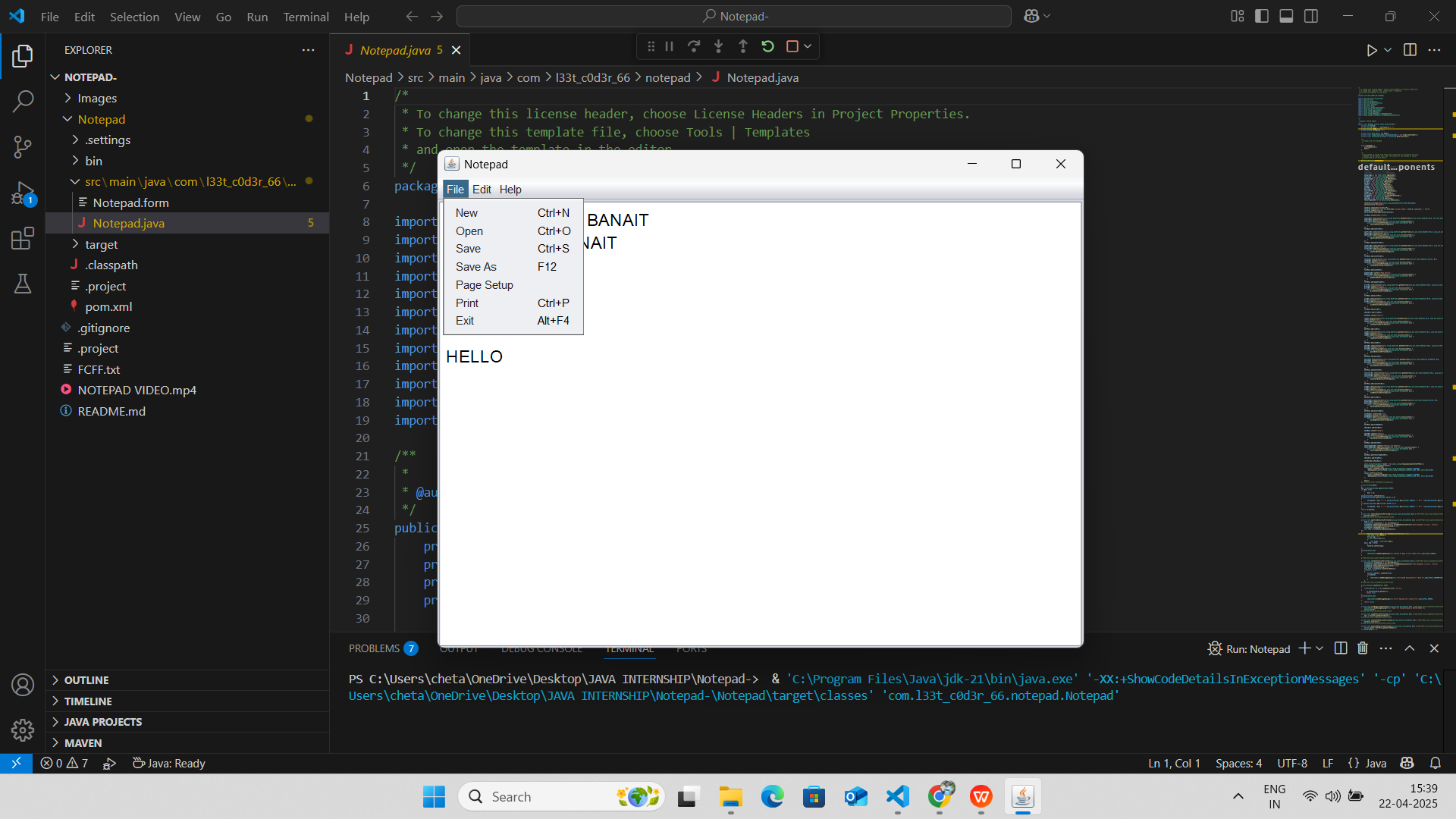
**User-Friendly Interface:**  
Built using Swing components like JTextArea, JMenuBar, JFileChooser, JOptionPane, and more for an interactive and intuitive UI.



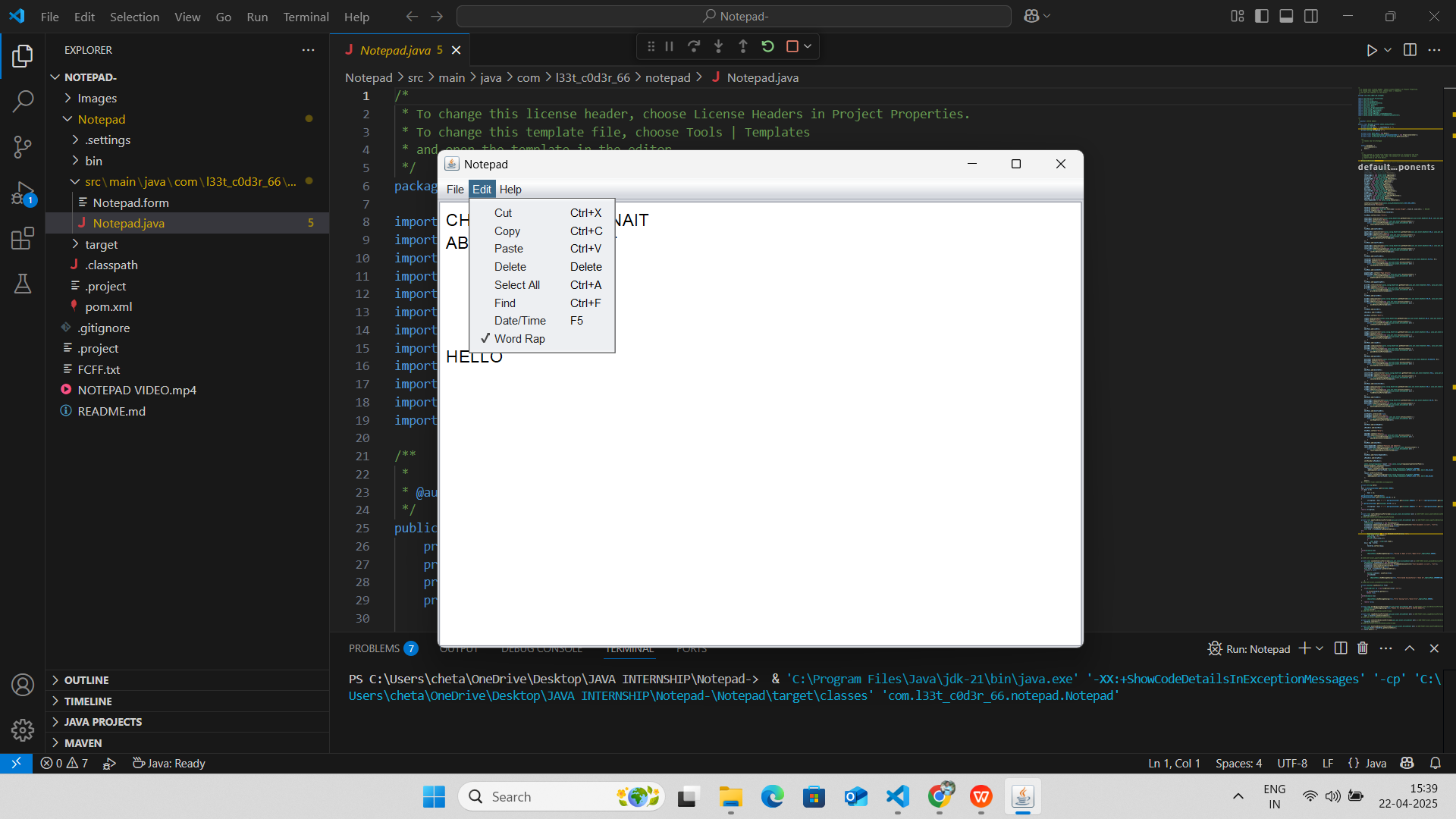
*Figure 1.:- Write a text and Print option*



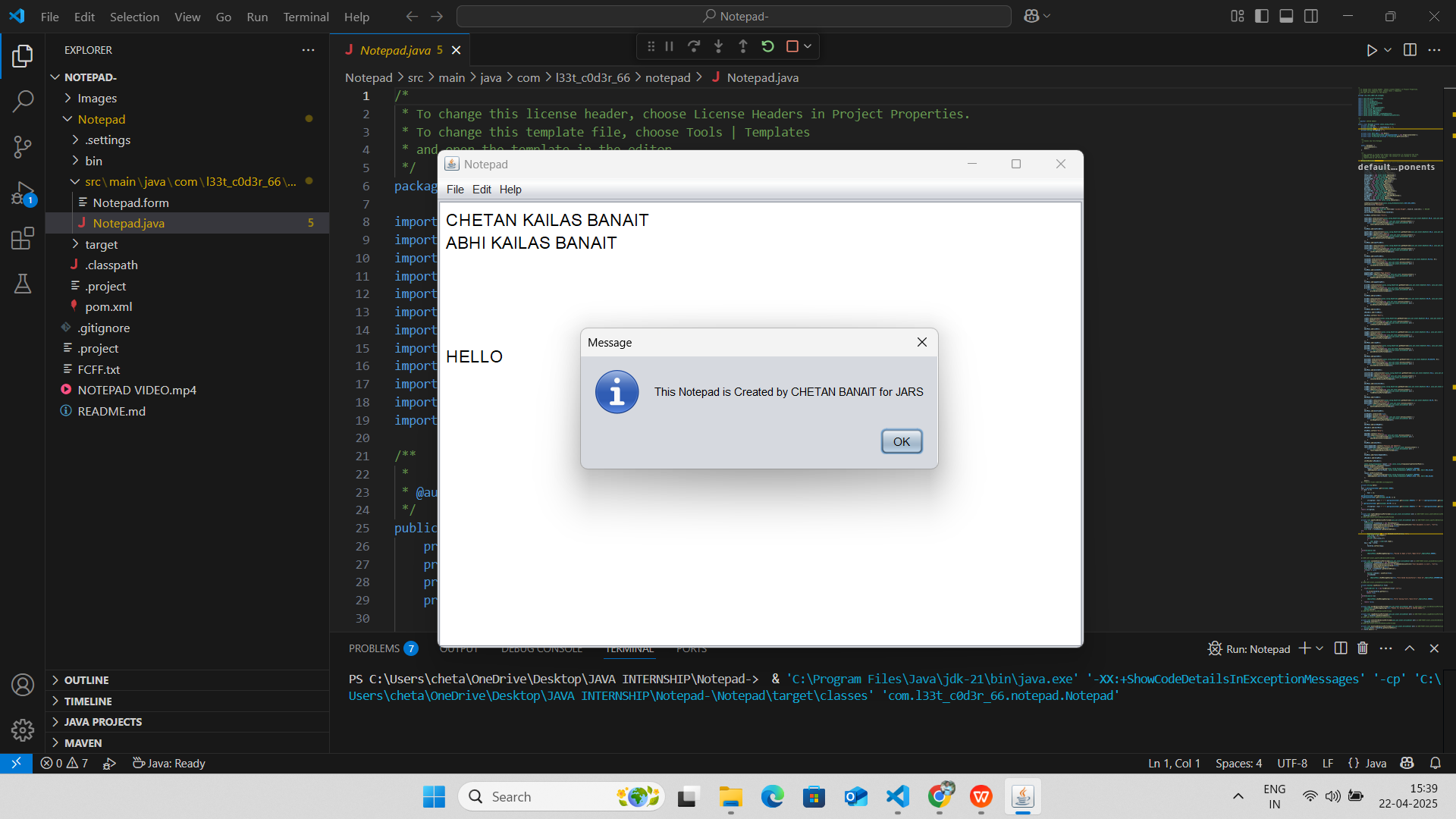
*Figure 2.:- Write a text and Save option*



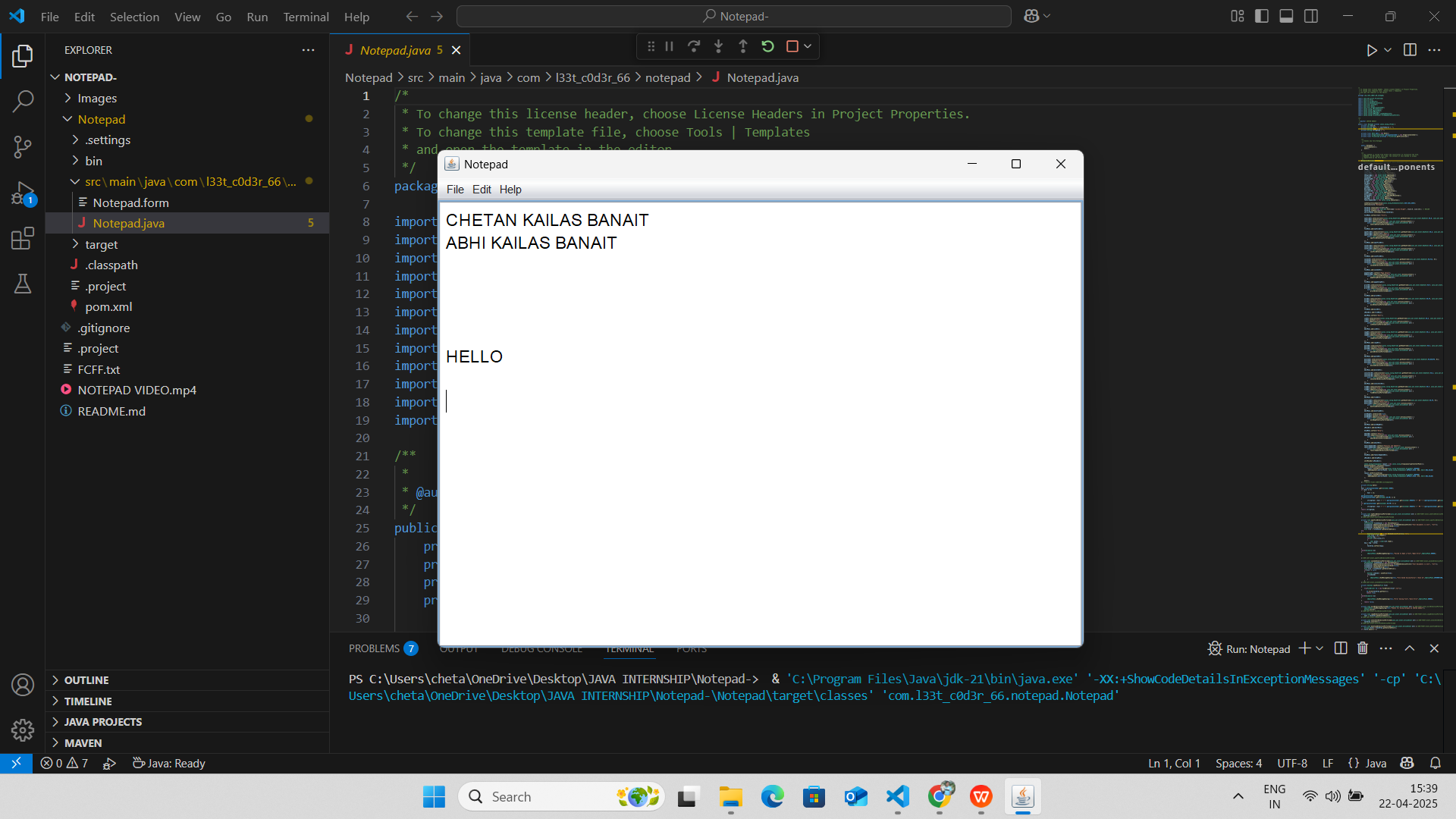
*Figure 3.:- File Menu Option*



*Figure 4.:- Edit Menu Option*



*Figure 5.:- File Error Massage*



*Figure 6.:- Write something on notepad and perform operations*

**Final Project:**

## ****Project Overview****

The **Notepad Clone** is a desktop-based Java application designed to provide users with a lightweight, easy-to-use text editing tool. Developed using **Core Java, AWT, and Swing**, the application mimics the functionality of the standard Windows Notepad, offering essential features such as creating, opening, editing, and saving text files. It serves as an ideal project for understanding GUI development, event handling, and file manipulation in Java, especially for students or beginners in desktop application development.

**Objective and Motivation**

The main objective of this project is to build a simple and functional text editor that enhances Java programming skills, especially in GUI and file I/O operations. The motivation came from the widespread use of Notepad-like tools for quick note-taking, code editing, and writing documents. By creating a Notepad Clone, I aimed to deepen my understanding of user interface design and reinforce the principles of modular and event-driven programming.

## ****Features Implemented****

* **Create New File**: Clear the current content to start a new document.
* **Open File**: Load .txt files from the local system using file chooser dialogs.
* **Save/Save As**: Save the current text to a file, with overwrite or rename options.
* **Cut, Copy, Paste**: Standard editing operations using clipboard support.
* **Find and Replace**: Locate specific words/phrases and optionally replace them.
* **Font Formatting**: Customize text font, style, size, and color.
* **Status Bar** (optional): Display line number, character count, etc.
* **Exit Confirmation**: Prompts to save unsaved changes before exiting.

The application includes input validations, exception handling, and clean UI navigation, ensuring a robust and smooth user experience.

## ****Frontend & Backend Technologies Used****

**Frontend**: Java Swing (JFrame, JMenuBar, JTextArea, JFileChooser)

**Backend**: Core Java (Event Handling, File I/O)

**IDE Used**: NetBeans / Eclipse / IntelliJ IDEA

**Development Tools**: Java SDK (JDK 8 or above)

## ****Data Handling Approach****

As a desktop-based application, the Notepad Clone does not use a database. Instead, it handles text data using:

FileReader and BufferedReader for reading files

FileWriter and BufferedWriter for saving content

Dialog boxes for user inputs and confirmations

This ensures fast performance and low memory usage, ideal for small-scale or personal use.

## ****APIs Created (Modular Java Methods)****

Although no external APIs are used, the system includes well-defined internal methods such as:

createNewFile()

openFile()

saveFile()

findText()

replaceText()

These modular methods improve code organization and follow the **Model-View-Controller (MVC)** design pattern where applicable.

## ****Deployment Details****

The project is packaged into an executable .jar file, making it easy to run across any platform with Java installed. It requires no external dependencies apart from the JDK. The application can be deployed on standalone systems, USB drives, or integrated into larger desktop suites. For future expansion, cloud save, theme support, and plugin extensions could be added.

**Learning Outcomes**

### ****Key Takeaways****

Developing the Notepad Clone project offered valuable insights into applying Core Java for building real-world desktop applications. Key learnings included mastering event-driven programming, understanding GUI design principles, and efficiently handling file operations using Java I/O streams. I strengthened my grasp on modular programming, which made the codebase easier to manage, debug, and extend. Additionally, this project helped me develop structured problem-solving skills and understand the critical role of user experience in software design.

### ****Tools Learned****

During the project, I became familiar with several tools and technologies that are widely used in industry:

* **Core Java** – for implementing the main application logic and structure.
* **AWT and Swing** – for creating and managing the Graphical User Interface.
* **Java I/O (FileReader, FileWriter)** – for reading and writing text files.
* **NetBeans / Eclipse / IntelliJ IDEA** – used as IDEs to develop, test, and debug the project.
* **Git** – for source code version control and managing iterative development.

These tools enhanced my coding efficiency and gave me a hands-on introduction to professional-grade development environments.

### ****Real-world Experience****

This project simulated many real-world software development challenges. I practiced gathering requirements, designing UI layouts, implementing features, and thoroughly testing for user errors and performance issues. Realistic situations such as handling unsaved file warnings, text formatting limitations, and multi-window handling deepened my technical understanding. Writing clean, maintainable, and well-commented code became a key focus, along with ensuring that the application responded gracefully to user actions. Overall, the experience significantly boosted my confidence to take on more complex development tasks in the future.

**Conclusion**

### ****Internship Impact****

This internship significantly contributed to both my technical growth and professional development. Working on the Notepad Clone project helped me apply my knowledge of Core Java in a real-world context. I gained hands-on experience with Java Swing, event-driven programming, and file handling through Java I/O, which enhanced my confidence in developing GUI-based desktop applications. The internship taught me the value of writing clean, maintainable code, managing features in a modular fashion, and handling common user interactions gracefully. I also improved in using tools like NetBeans, IntelliJ IDEA, and Git for efficient development and version control. Most importantly, I learned how to approach problems systematically, test thoroughly, and document clearly — skills that are crucial for a career in software development.

### ****Future Scope****

The **Notepad Clone** project forms a solid base that can be further enhanced with more advanced features. Future improvements could include:

**Syntax highlighting** for various programming languages.

**Tab-based document editing** for handling multiple files.

**Find and Replace** functionality with regular expression support.

**Spell check integration**.

**Autosave and file recovery features**.

**Cloud file sync** using APIs such as Google Drive or Dropbox.

Conversion into a **cross-platform application** using frameworks like JavaFX or Electron.

Additionally, this project can be expanded into a **collaborative text editor** with real-time editing using sockets or web-based technologies. The internship provided the confidence and foundational knowledge necessary to pursue these improvements and evolve the project into a more sophisticated tool.

**Appendix**

### ****1. Code Snippets****

Below are some key code snippets that illustrate core functionalities of the Notepad Clone:

**a. File Opening Functionality:**

java

CopyEdit

public void openFile() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(this);

if (option == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

try (BufferedReader br = new BufferedReader(new FileReader(file))) {

textArea.read(br, null);

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Error Opening File: " + e.getMessage());

}

}

}

**b. Saving File Content:**

java

CopyEdit

public void saveFile() {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(this);

if (option == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {

textArea.write(bw);

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Error Saving File: " + e.getMessage());

}

}

}

**c. Word Wrap Toggle:**

java

CopyEdit

public void toggleWordWrap() {

isWordWrapOn = !isWordWrapOn;

textArea.setLineWrap(isWordWrapOn);

textArea.setWrapStyleWord(isWordWrapOn);

}

**d. Exit Confirmation:**

java

CopyEdit

public void exitApplication() {

int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to exit?", "Exit", JOptionPane.YES\_NO\_OPTION);

if (confirm == JOptionPane.YES\_OPTION) {

System.exit(0);

}

}

### ****2. GitHub Links****

You can find the complete source code and project files on GitHub:

### ****3. References****

· **Java™ Platform, Standard Edition Documentation – Oracle**  
[https://docs.oracle.com/javase/8/docs/](https://docs.oracle.com/javase/8/docs/" \t "_new)  
– Official documentation covering the Java SE API used for building the Notepad Clone.

· **Java Swing Tutorial – Oracle**  
[https://docs.oracle.com/javase/tutorial/uiswing/](https://docs.oracle.com/javase/tutorial/uiswing/" \t "_new)  
– Comprehensive tutorial on building GUIs with Swing, essential for the Notepad interface.

· **JFileChooser API – Oracle Docs**  
[https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html](https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html" \t "_new)  
– Used for file selection dialogs in the open/save functionalities.

· **Java IO Package – Oracle Docs**  
[https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html](https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html" \t "_new)  
– For reading from and writing to files, enabling the file management features in the Notepad.

· **Git – Version Control System**  
[https://git-scm.com/doc](https://git-scm.com/doc" \t "_new)  
– Used to track and manage code changes during development.

· **Stack Overflow Community**  
[https://stackoverflow.com/](https://stackoverflow.com/" \t "_new)  
– Helpful for resolving errors, improving features, and understanding best practices in Java.