Java PROGRAMMING

Project Name	JAVA Programming
Project Institute	CodTech IT Solutions
Task Name	CREATE A JAVA PROGRAM TO READ, WRITE, AND MODIFY TEXT FILES. DELIVERABLE: A SCRIPT DEMONSTRATING FILE OPERATIONS WITH CLEAR DOCUMENTATION
Prepared By	Janani Sri.P

Rev No	Date	Task Description	Prepared By
Rev 00	22.07.02025	A simple menu-driven Java program to read,write,and modify text files 📁 Java File Operations This is a simple Java console-based project demonstrating file handling operations:	Janani Sri.P

Task Name

CREATE A JAVA PROGRAM TO READ, WRITE, AND MODIFY TEXT FILES.
DELIVERABLE: A SCRIPT DEMONSTRATING FILE OPERATIONS WITH CLEAR DOCUMENTATION

Details

A simple menu-driven Java program to read,write,and modify text files 📁 Java File Operations

This is a simple Java console-based project demonstrating file handling operations

Packages required:

- java.io or java.nio.file.

Test Scope(Description)

- **Write (Overwrite)** – Replaces the entire content of the file with new text provided by the user.
- **Append** – Adds new text to the end of the file without deleting the existing content.
- **Read** – Displays the current contents of the file on the console.
- **Exit** – Safely terminates the program.

Summary of required things

Java PROGRAMMING

Operation	Java Class(es) Needed
Write	FileWriter
Append	FileWriter(true)
Read	FileReader, BufferedReader
Exit	System.exit()
Exception Handling	try-catch with IOException

Tools,Versions and its used

---

1 .Java Development Kit (JDK)

- **Required Version:** JDK 8 or above (JDK 17+ is recommended for long-term support).
- **Used For:**
    - Compiling and running the Java program.
    - Provides core Java libraries like java.io.

Download: https://www.oracle.com/java/technologies/javase-downloads.html
Or use OpenJDK: https://jdk.java.net/

2. Text File (For File Operations)

- **Required File:** A text file like example.txt
- **Used For:**
    - Reading existing content.
    - Writing or appending user input.
- You can let the program create this file dynamically, or create it manually in your project folder.

Libraries:

1.File Writer
2.File reader
3.Bufferedreader
4.IOExecptions
5.Scanner

Base Code:

This is the java code to write,read,append and exit for the program

Java PROGRAMMING

Program

```
import java.io.*;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      String fileName = "example.txt";
      Scanner scanner = new Scanner(System.in);
      int choice;

      do {
         System.out.println("\n📂 File Operations Menu");
         System.out.println("1. Write to File (Overwrite)");
         System.out.println("2. Read from File");
         System.out.println("3. Append to File");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");
         choice = scanner.nextInt();
         scanner.nextLine(); // Clear newline

         switch (choice) {
            case 1:
               // Write to file
               try {
                  FileWriter writer = new FileWriter(fileName);
                  System.out.println("✍️ Enter text to write (type 'exit' to stop):");
                  while (true) {
                     String input = scanner.nextLine();
                     if (input.equalsIgnoreCase("exit")) break;
                     writer.write(input + "\n");
                  }
```

```
            writer.close();
            System.out.println("✅ File written successfully.");
        } catch (IOException e) {
            System.out.println("❌ Error writing to file.");
            e.printStackTrace();
        }
        break;

    case 2:
        // Read file
        try {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));
            String line;
            System.out.println("\n📄 File Content:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            System.out.println("❌ Error reading file.");
            e.printStackTrace();
        }
        break;

    case 3:
        // Append to file
        try {
            FileWriter writer = new FileWriter(fileName, true); // append mode
            System.out.println("➕ Enter text to append (type 'exit' to stop):");
            while (true) {
```

```
                String input = scanner.nextLine();
                if (input.equalsIgnoreCase("exit")) break;
                writer.write(input + "\n");
            }
            writer.close();
            System.out.println("✅ Content appended successfully.");
        } catch (IOException e) {
            System.out.println("❌ Error appending to file.");
            e.printStackTrace();
        }
        break;

      case 4:
        System.out.println("👋 Exiting program.");
        break;

      default:
        System.out.println("⚠️ Invalid choice. Try again.");
    }
  } while (choice != 4);

  scanner.close();
 }
}
```

Alpha Stage

Iteration-1

Date and Time

22/7/25,3:00pm

Observation data

1.Java Provides Built-in Support for File Handling:

- Java's java.io and java.util packages include all the necessary classes to perform file operations without needing any third-party libraries.

2. **FileWriter and FileReader Handle Character Streams:**
    - FileWriter is used for writing characters to a file, while FileReader is used for reading characters from a file.
    - When FileWriter is used without the append flag, it overwrites the file; with true, it appends data.
3. **BufferedReader Improves Reading Efficiency:**
    - BufferedReader reads text line by line, making it more efficient than reading one character at a time.
4. **Scanner Simplifies Console Input:**
    - Scanner class from java.util is used to read user input for dynamic file operations.
5. **Exception Handling is Essential:**
    - File operations are prone to runtime errors (e.g., file not found, permission denied), so IOException is handled using try-catch blocks.
6. **Menu-Driven Programs Improve Usability:**
    - A simple menu (e.g., 1. Write, 2. Append, 3. Read, 4. Exit) makes it user-friendly to perform various file operations.
7. **File Creation is Handled Automatically:**
    - If the specified file does not exist, Java automatically creates it when writing or appending for the first time.
8. **Exiting the Program Gracefully:**
    - System.exit(0); allows the program to terminate cleanly when the user selects the exit option.

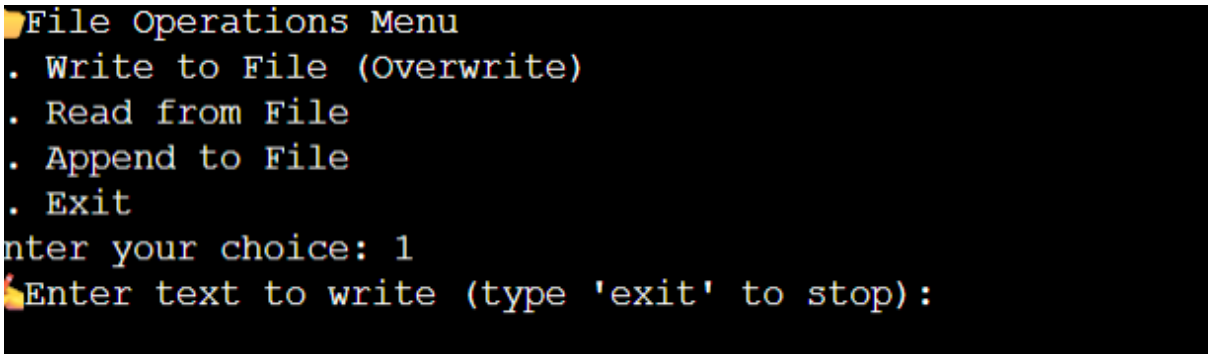
How It Works

- Type input to write or append
- View file content anytime
- Enter exit to stop writing/appending

In code editor it will look like this when we run the code and its ask input by entering the choices ….its given belown

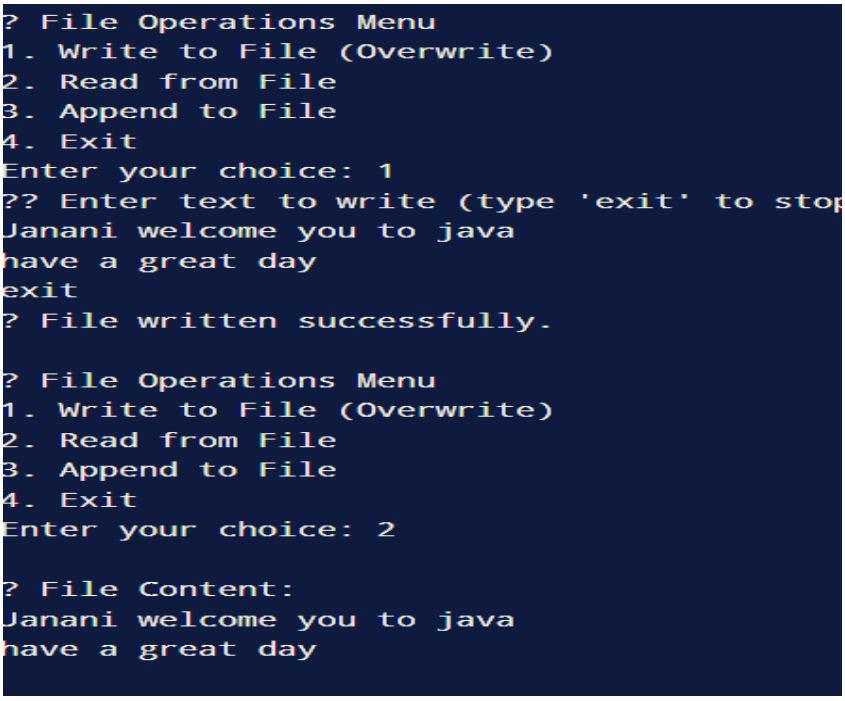
Java PROGRAMMING

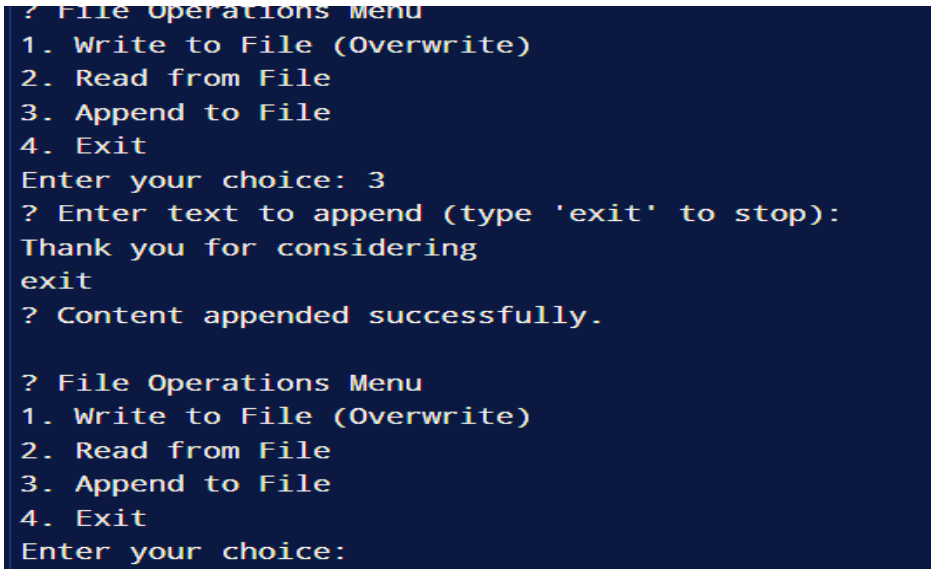
Inputs:



Java PROGRAMMING

Output:





Results

Thus the code has been successfully verified and got the output.

To check this on code on github link : https://github.com/Janani-6382/Java--File-Operations

Conclusion

This program demonstrates the fundamental file handling capabilities provided by Java's standard libraries. By using built-in classes such as FileWriter, FileReader, BufferedReader, and

Scanner, users can efficiently perform essential operations like writing, appending, and reading from a file. Exception handling ensures that the application runs reliably even when errors occur, such as missing files or I/O failures. The menu-driven interface enhances usability, making the program suitable for both beginners and practical real-world scenarios.