

LockedMe

This specification document contains sections for:

- Sprint Planning
- Core concepts used in project
- Flow of the Application.
- Demonstrating the product capabilities, appearance, and user interactions
- Unique Selling Points of the Application

The code repository for this project can be found at the below link:

<https://github.com/shrawanthakur16/SimpliLearnProjects/tree/main/src/com/LockedMe>

The project is developed by Shrawan Kumar Thakur.

Sprints planning and Task completion

The project is planned and completed in 2 sprints. The tasks assumed to be completed in the **first sprint** are:

- Creating the flowchart of the application
- Writing java code for welcome screen, file creation and file searching.
- Testing the program with different kinds of input to see output

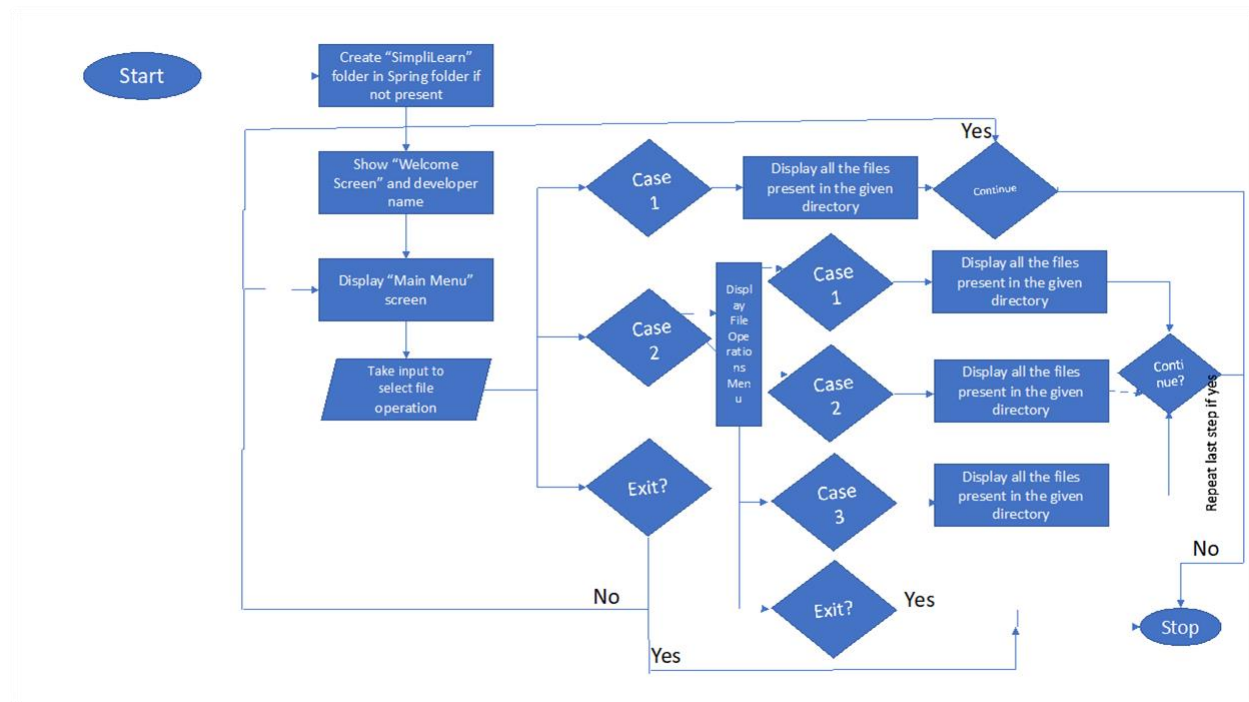
The tasks assumed to be completed in the **second and final sprint** are:

- Writing java code for listing all the files created or present in the folder already and deleting a file.
- Testing the program with different kinds of input to see the output
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Java concepts used in the project

File Handling, Exception Handling, Recursion, Collections like List and ArrayList, Sorting, Searching and Stream API.

Flowchart of the Application



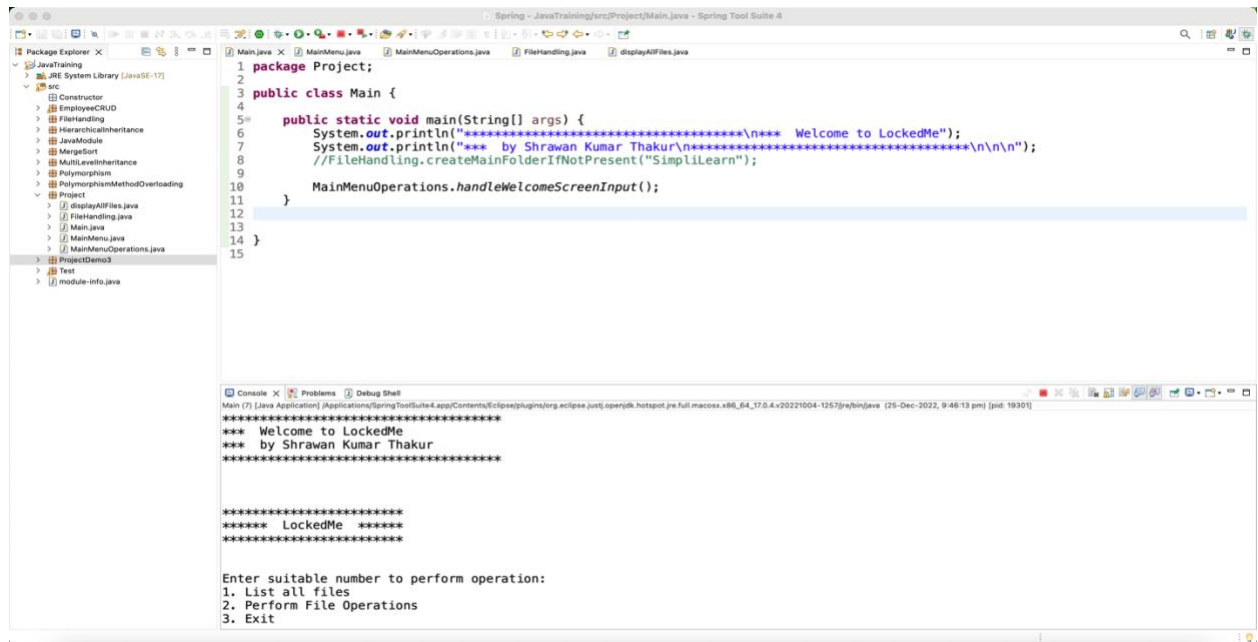
Link of the Image:

<https://github.com/shrawanthakur16/SimpliLearnProjects/blob/5632dc108812c7a37b30e56b85a788020c609166/Specifications%20Documents/LockedMe%20Flowchart.pdf>

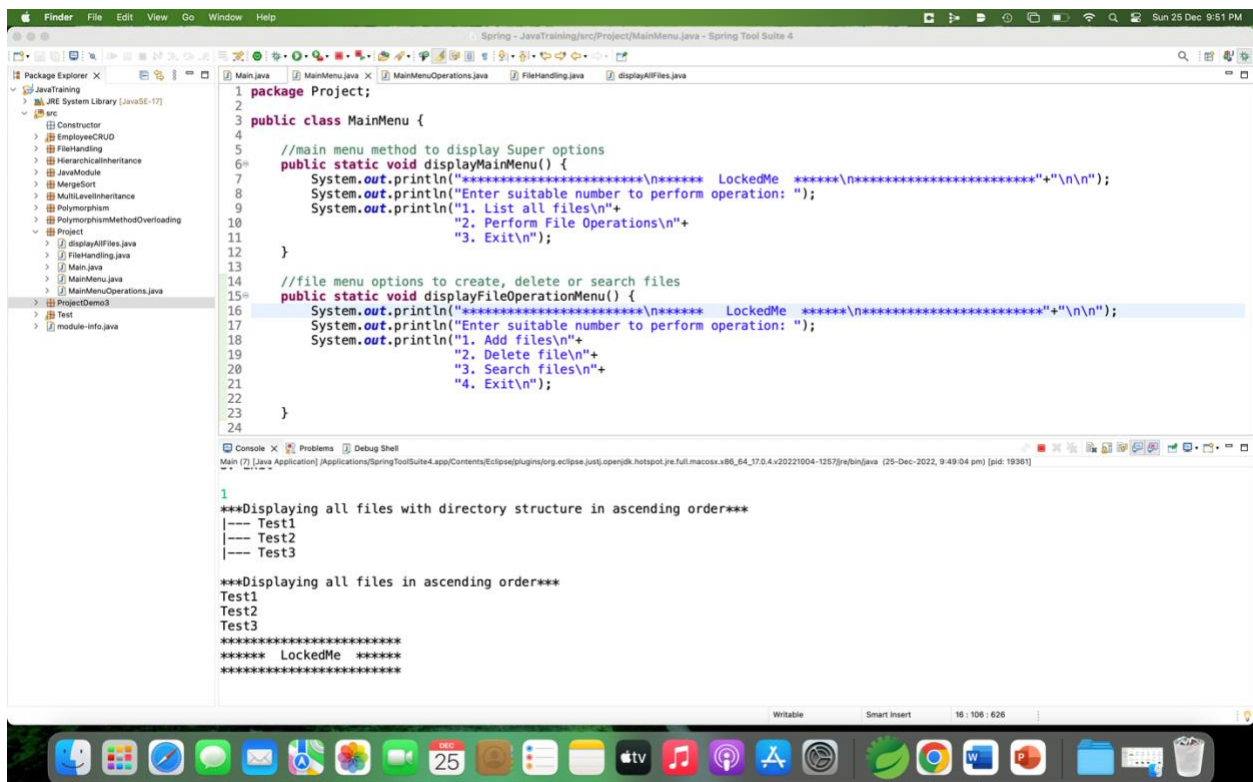
Demonstrating the product capabilities, appearance, and user interactions

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

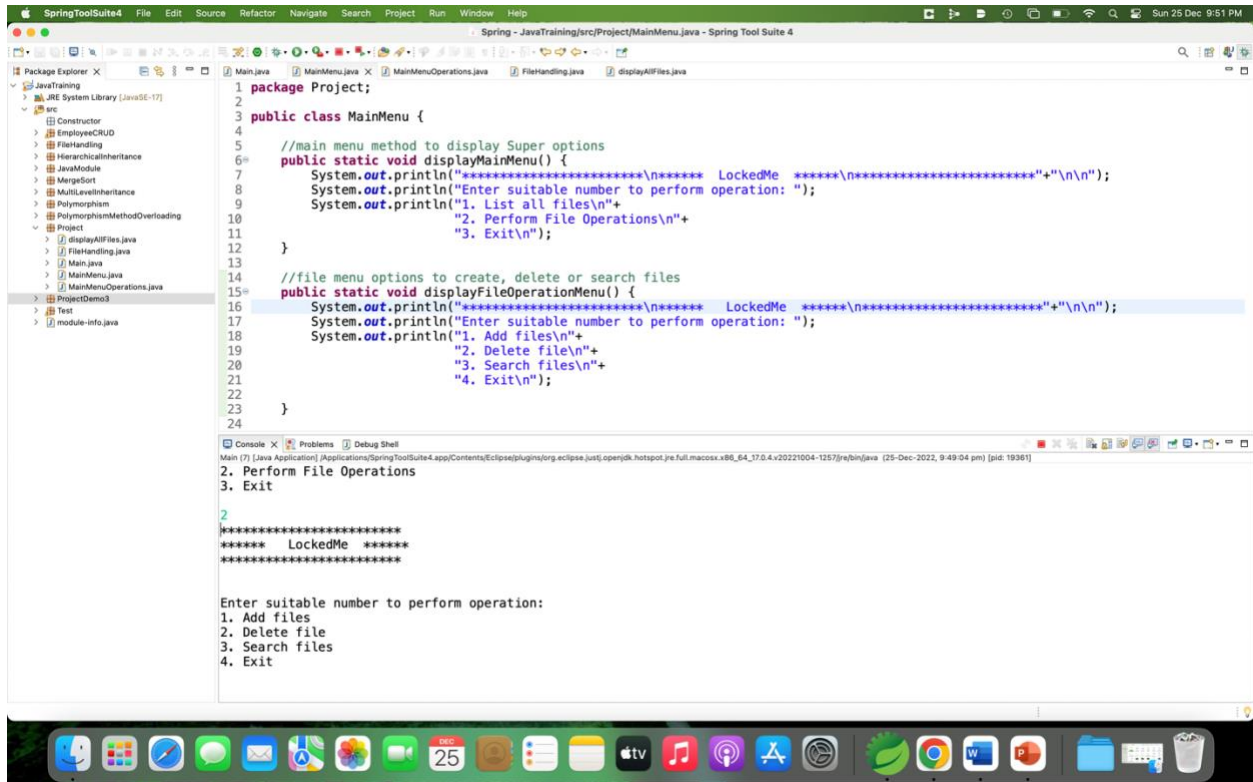
1. Welcome screen



2. List all Files



3. Main Menu Options Screen



The screenshot shows the Spring Tool Suite 4 IDE with the `MainMenu.java` file open. The code defines a `MainMenu` class with two static methods: `displayMainMenu()` and `displayFileOperationMenu()`. The console output shows the program running, displaying the main menu options and the file operation menu options. The user has entered '2' to select 'Perform File Operations'.

```
package Project;

public class MainMenu {

    //main menu method to display Super options
    public static void displayMainMenu() {
        System.out.println("*****\n***** LockedMe *****\n*****\n*****");
        System.out.println("Enter suitable number to perform operation: ");
        System.out.println("1. List all files\n"+
            "2. Perform File Operations\n"+
            "3. Exit\n");
    }

    //file menu options to create, delete or search files
    public static void displayFileOperationMenu() {
        System.out.println("*****\n***** LockedMe *****\n*****\n*****");
        System.out.println("Enter suitable number to perform operation: ");
        System.out.println("1. Add files\n"+
            "2. Delete file\n"+
            "3. Search files\n"+
            "4. Exit\n");
    }
}
```

Console Output:

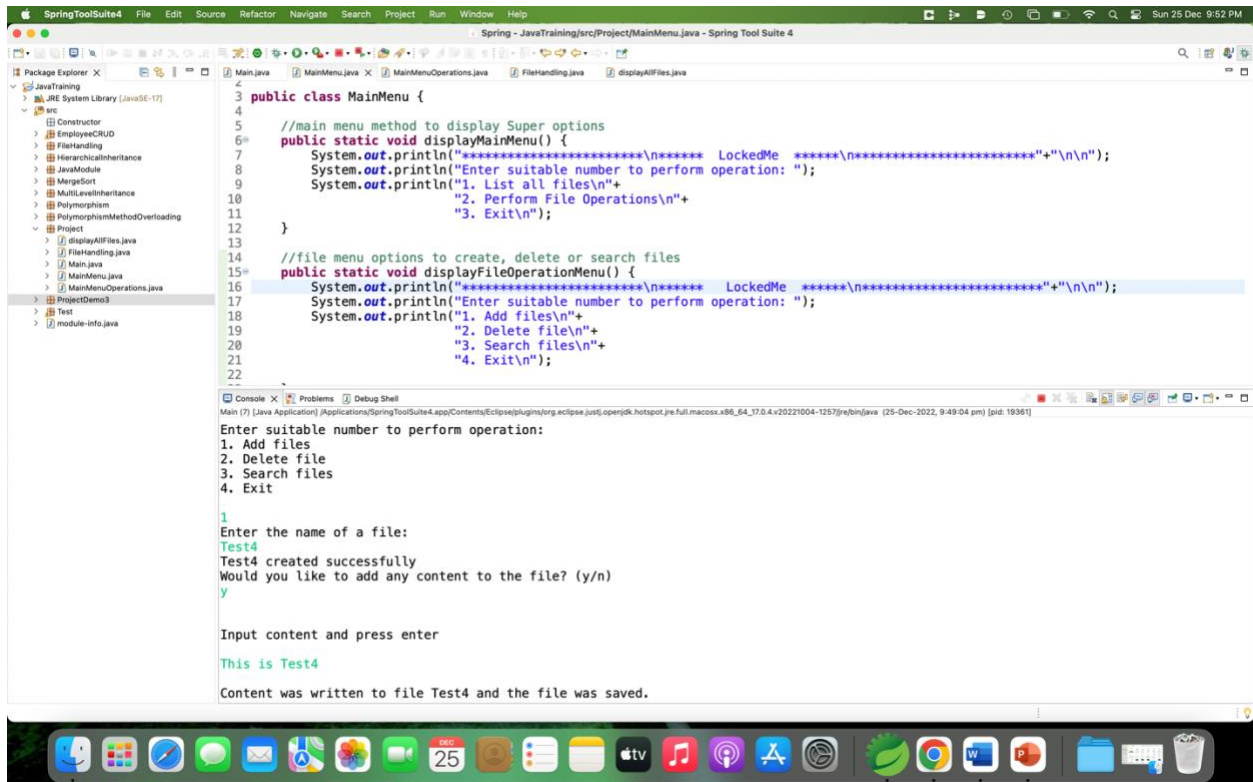
```
Main (7) [Java Application] Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86_64_17.0.4.v20221004-1257/jre/bin/java (25-Dec-2022, 9:49:04 pm) [pid: 19361]

2. Perform File Operations
3. Exit

2
*****\n***** LockedMe *****\n*****\n*****

Enter suitable number to perform operation:
1. Add files
2. Delete file
3. Search files
4. Exit
```

4. Creating file



The screenshot shows the Spring Tool Suite 4 IDE with the `MainMenu.java` file open. The code is the same as in the previous screenshot. The console output shows the program running, displaying the main menu options and the file operation menu options. The user has entered '1' to select 'Add files', then entered 'Test4' as the filename. The console output shows that the file was created successfully and the user was asked if they wanted to add content to the file. The user entered 'y', and the console output shows that the content 'This is Test4' was written to the file.

```
package Project;

public class MainMenu {

    //main menu method to display Super options
    public static void displayMainMenu() {
        System.out.println("*****\n***** LockedMe *****\n*****\n*****");
        System.out.println("Enter suitable number to perform operation: ");
        System.out.println("1. List all files\n"+
            "2. Perform File Operations\n"+
            "3. Exit\n");
    }

    //file menu options to create, delete or search files
    public static void displayFileOperationMenu() {
        System.out.println("*****\n***** LockedMe *****\n*****\n*****");
        System.out.println("Enter suitable number to perform operation: ");
        System.out.println("1. Add files\n"+
            "2. Delete file\n"+
            "3. Search files\n"+
            "4. Exit\n");
    }
}
```

Console Output:

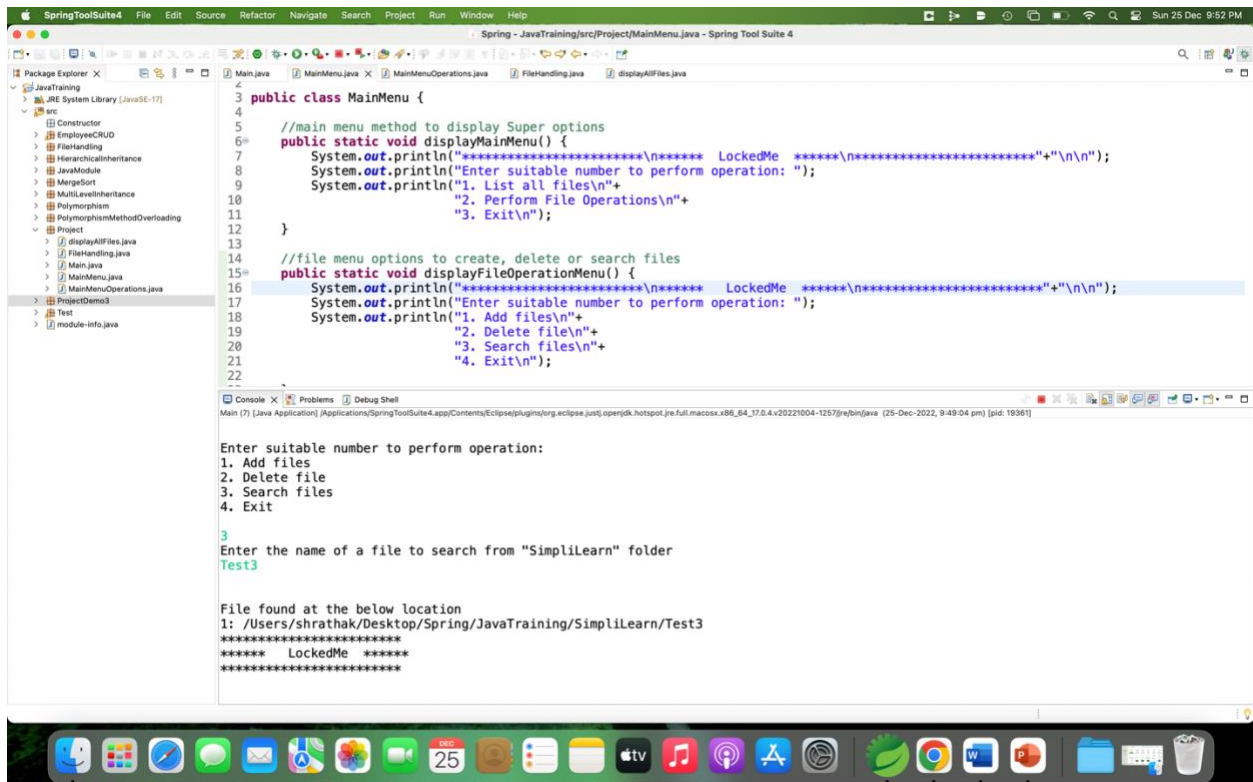
```
Main (7) [Java Application] Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86_64_17.0.4.v20221004-1257/jre/bin/java (25-Dec-2022, 9:49:04 pm) [pid: 19361]

Enter suitable number to perform operation:
1. Add files
2. Delete file
3. Search files
4. Exit

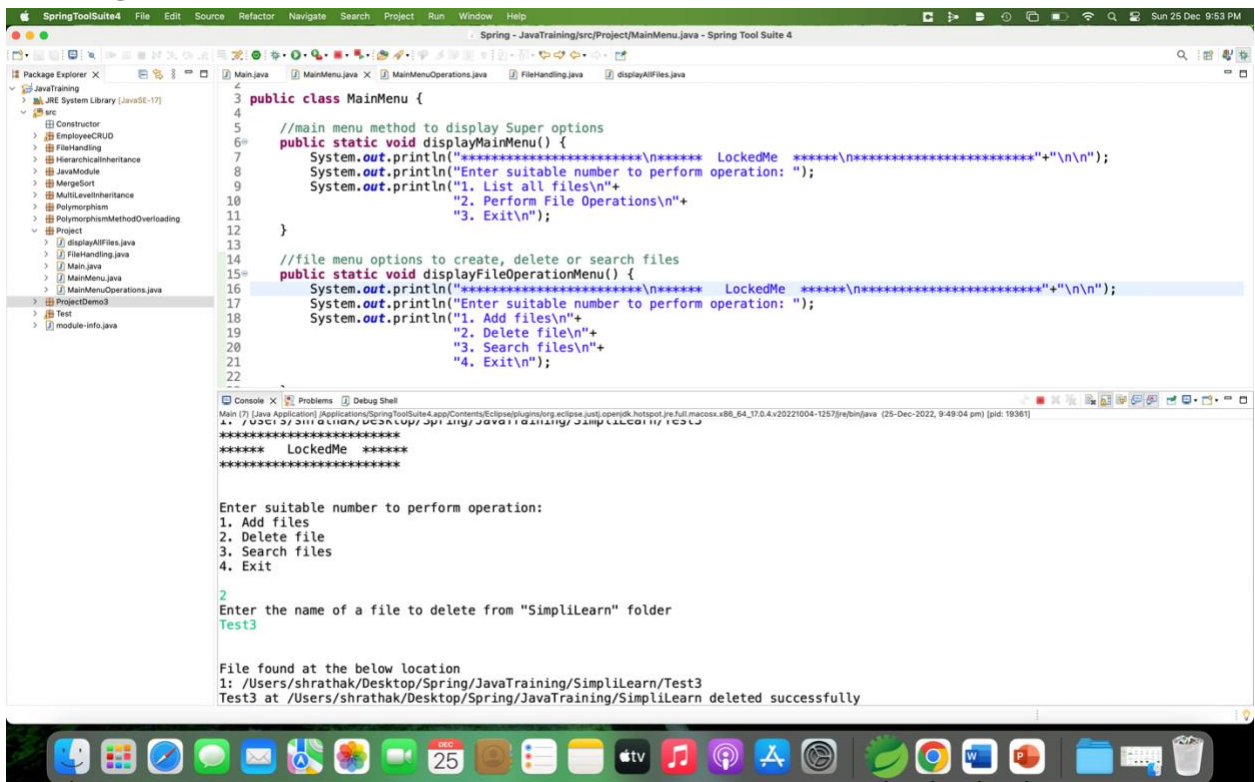
1
Enter the name of a file:
Test4
Test4 created successfully
Would you like to add any content to the file? (y/n)
y

Input content and press enter
This is Test4
Content was written to file Test4 and the file was saved.
```

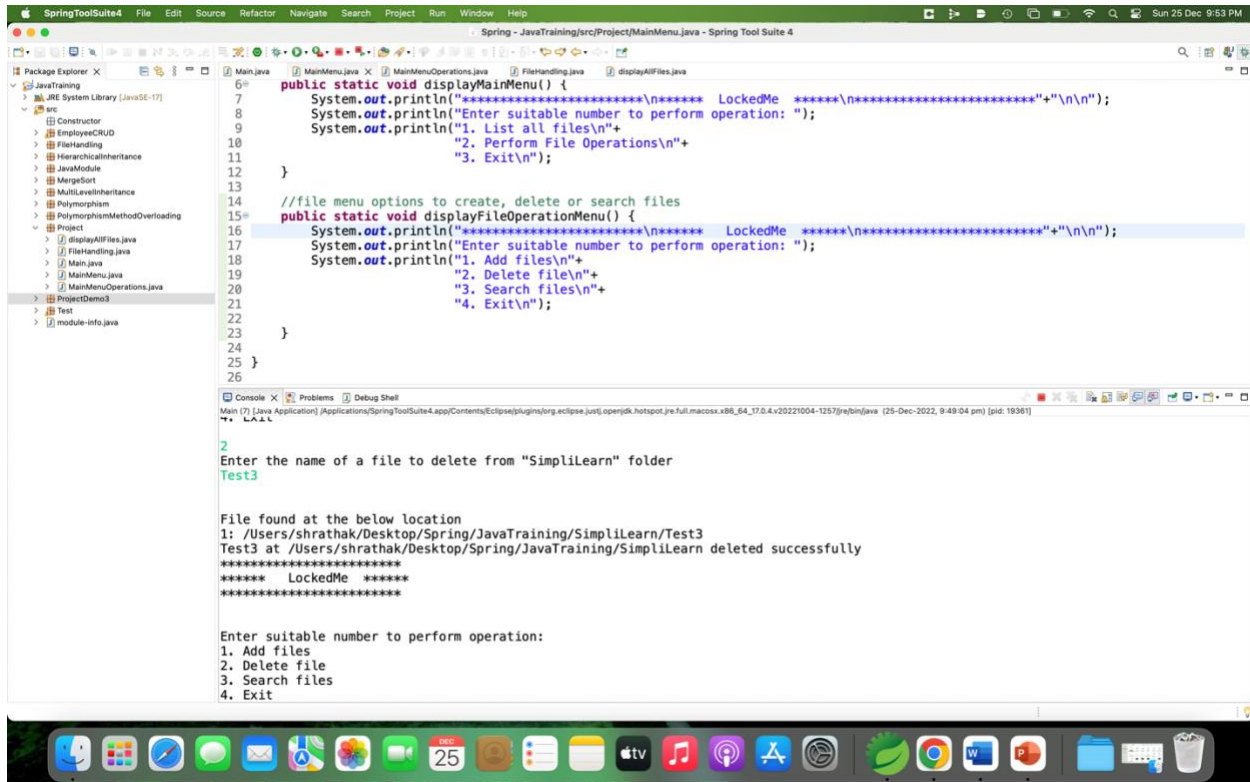
5. Searching a file



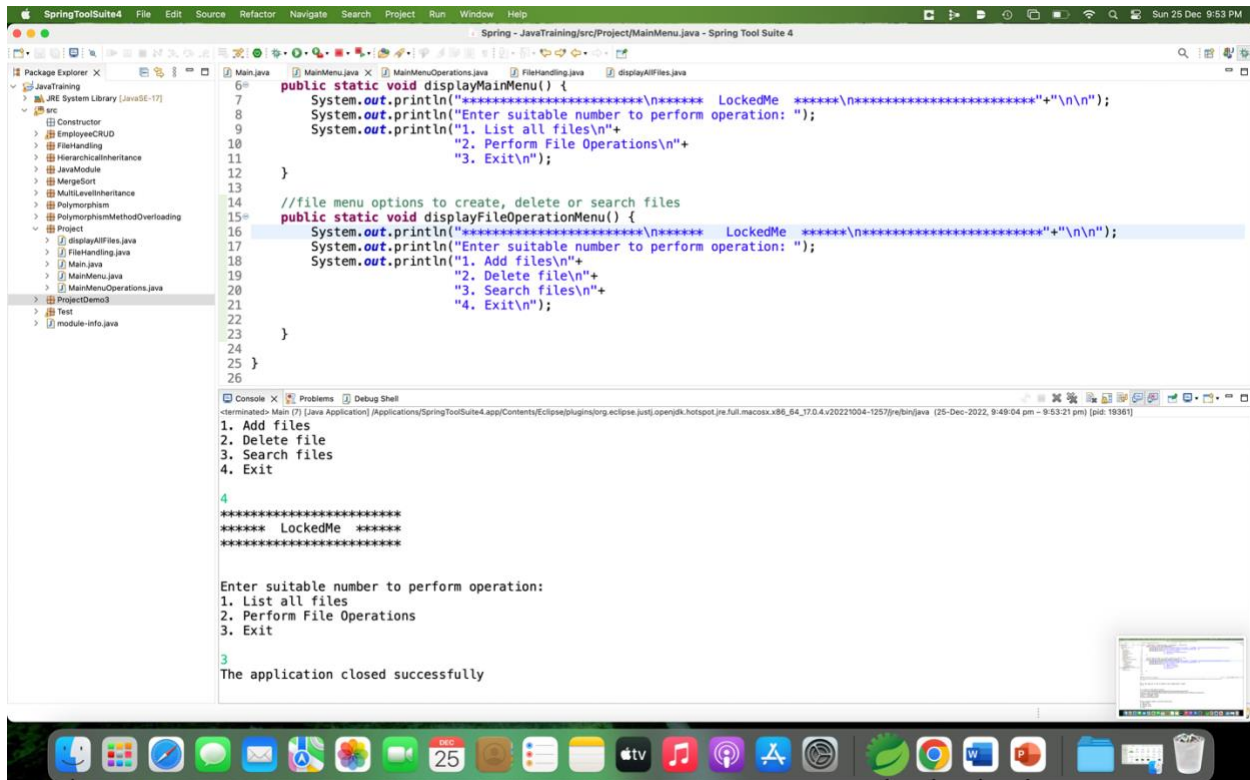
6. Deleting a file



7. Exiting from the File options menu to Main Menu



8. Exiting the application



Unique Selling Point: The application can be scaled into larger application as it designed with modularity in mind and is exactly what a file manager interface should look like. The application is very simple, easy to understand, modular and has much less hardcode.