LockedMe.com – Virtual Key for Repositories

The code for this project is hosted at

https://github.com/keerthana-777/LockedMe.git

LockedMe.com is a user-friendly application designed to provide the file handling operations, taking inputs from the user by performing operations as per user needs.
The project is developed by **KEERTHANA BANDARI**.

This document contains sections for:

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

Sprints planning and Task completion:

The project is planned to be completed in 3 sprints and the duration is 3 weeks. Each sprint is planned to be completed within a week.

Tasks assumed to be completed in the 1st sprint is:

- Generating the flow of the application through flow chart.
- Initializing git repository to track changes in order to progress the development.
- Writing the Java program in eclipse to fulfill the requirements of the project.
- Testing the Java code with various kinds of User input and tracking output.
- Delivering a potentially releasable product.

Tasks assumed to be completed in the 2nd sprint is:

- Developing the additional code/features specified by the user.

- Testing the Java code against various User inputs and tracking output.
- Delivering a potentially releasable product.

Tasks assumed to be completed in the 3rd sprint is:

- Developing the additional code/features specified by the user.
- Connecting the local repository to remote and pushing the code to GitHub.
- Making this specification document to emphasize the application capabilities, representation, and user interactions.
- Delivering a potentially releasable high-end quality product.

Core concepts used in project:

➢ **JAVA:**

- java has been one of the most popular programming languages for many years.
- Java is Object Oriented. However, it is not considered as pure object-oriented as it provides support for primitive data types (like int, char, etc)
- The Java codes are first compiled into byte code (machine-independent code). Then the byte code runs on **J**ava **V**irtual **M**achine (JVM) regardless of the underlying architecture.

➢ **Collections framework:**

- The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

- **File Handling:**
  - File handling in Java is defined as reading and writing data to a file. The particular file class from the package called **java.io** allows us to handle and work with different formats of files.
  - Thus, if we want to use a file class, we need to create an object of that particular class and should specify the filename or directory name.
- **Sorting:**
  - Sorting is a class of algorithms that are tasked with rearranging the positions of elements of an array such that all of its elements are either in ascending or descending order.
  - A good sorting algorithm also needs to ensure that elements having the same value don't change their locations in the sorted array.
  - Sorting is necessary for getting a concrete understanding of data structures and algorithms.

- **Recursion:**

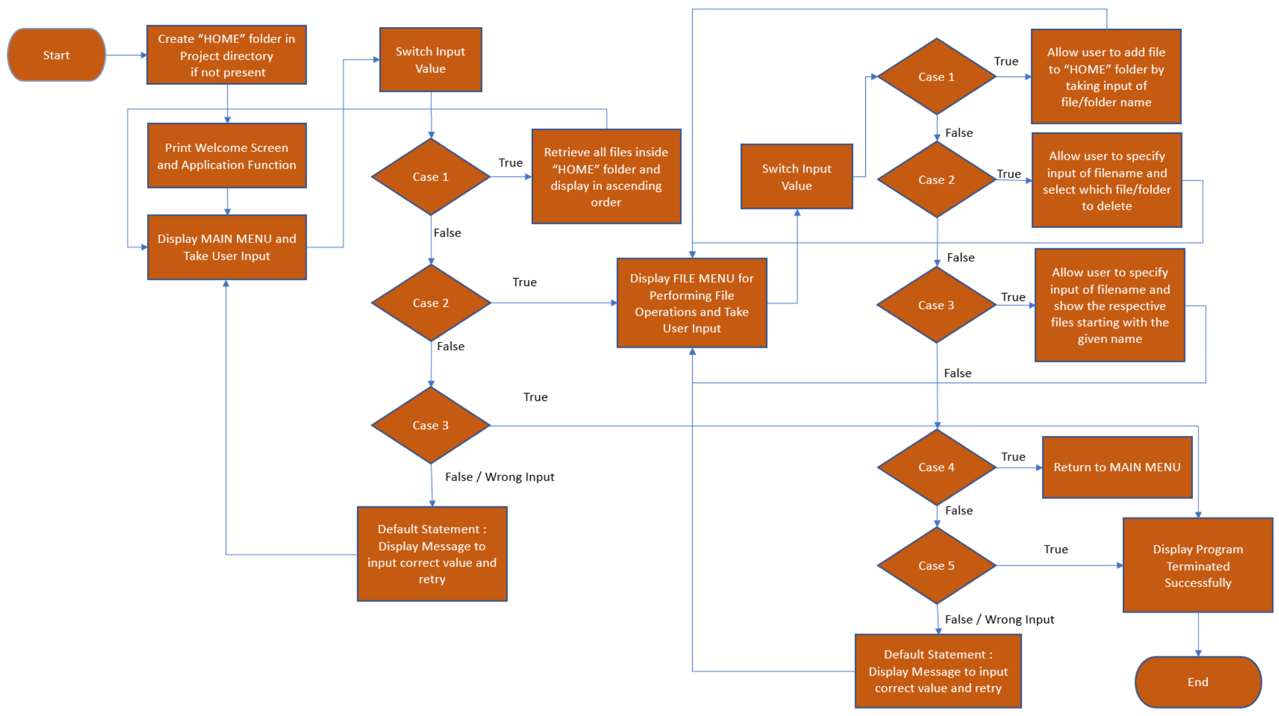
  - The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.
  - Using recursive algorithm, certain problems can be solved quite easily.

- **Exception Handling**:
  - The **Exception Handling** in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
  - This can be implemented using try, throw and catch blocks.

- **Streams API:**

- Stream operations are divided into intermediate and terminal operations and are combined form stream pipelines.
- A stream pipeline consists of a source (such as a Collection, an array, a generator function, or an I/O channel), followed by zero or more intermediate operations such as Stream.filter or Stream.map; and a terminal operation such as Stream.forEach or Stream.reduce.

Flow of the Application:


Start
Create "HOME" folder in Project directory if not present
Print Welcome Screen and Application Function
Display MAIN MENU and Take User Input
Switch Input Value
Case 1
True
Retrieve all files inside "HOME" folder and display in ascending order
False
Case 2
True
Display FILE MENU for Performing File Operations and Take User Input
False
Case 3
True
False / Wrong Input
Default Statement : Display Message to input correct value and retry
Switch Input Value
Case 1
True
Allow user to add file to "HOME" folder by taking input of file/folder name
False
Case 2
True
Allow user to specify input of filename and select which file/folder to delete
False
Case 3
True
Allow user to specify input of filename and show the respective files starting with the given name
False
Case 4
True
Return to MAIN MENU
False
Case 5
True
Display Program Terminated Successfully
False / Wrong Input
Default Statement : Display Message to input correct value and retry
End

Flow of Application.pptx

Demonstrating the product appearances, capabilities, and user interactions:

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

1. Creating the project in Eclipse
2. Writing a program in Java for the entry point of the application (**LockedMeMain.java**)
3. Writing a program in Java to display Menu options available for the user (**MenuOptions.java**)
4. Writing a program in Java to handle Menu options selected by user (**HandleOptions.java**)
5. Writing a program in Java to perform the File operations as specified by user (**FileOperations.java**)
6. Pushing the code to GitHub repository

Step 1: Creating a new project in Eclipse:

- Open Eclipse
- Go to File -> New -> Project -> Java Project -> Next.
- Type in any project name and click on "Finish."
- Select your project and go to File -> New -> Class.
- Enter **LockedMe** in any class name, check the checkbox "public static void main(String[] args)", and click on "Finish."

Step 2: Writing a program in Java for the entry point of the application (**LockedMeMain.java**):

```java
package com.lockedme;

public class LockedMeMain {
    public static void main(String[] args) {

        // Create "HOME" folder if not present in current folder
structure
        FileOperations.createMainFolder("HOME");

        MenuOptions.printWelcomeScreen("LockedMe", "KEERTHANA
BANDARI");

        HandleOptions.handleWelcomeScreen();
    }


}
```

Step 3: Writing a program in Java to display Menu options available for the user (**MenuOptions.java**):

- Select your project and go to File -> New -> Class.
- Enter **MenuOptions** in class name and click on “Finish.”
- **MenuOptions** consists methods for -:

3.1.Displaying Welcome Screen

3.2.Displaying **MAIN MENU**(Initial menu)

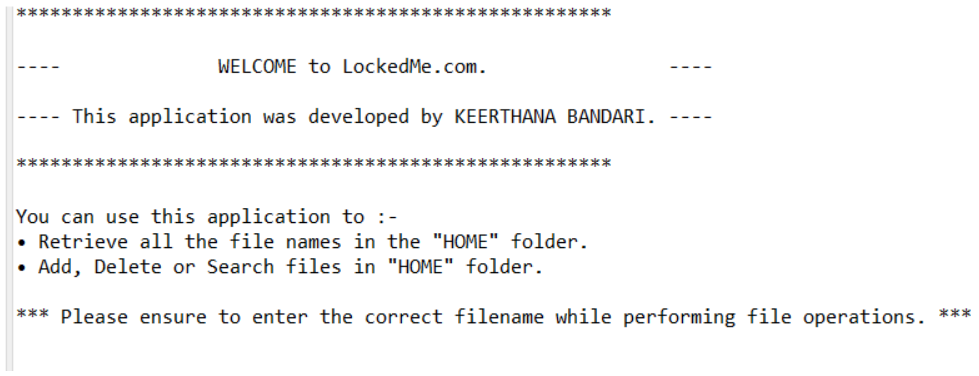
3.3.Displaying **FILE MENU**(Secondary menu) for File Operations available

Step 3.1: Writing method to display Welcome Screen:

```
public static void printWelcomeScreen(String appName, String
developerName) {
        String companyInfo =
String.format("*****************************************************\n
\n"
                + "----                WELCOME to %s.com.
----\n\n" + "---- This application was developed by %s. ----\n\n"
                +
"*****************************************************\n", appName,
developerName);
```

```
		String appFunction = "You can use this application to :-\n"
				+ "• Retrieve all the file names in the \"HOME\"
folder.\n"
				+ "• Add, Delete or Search files in \"HOME\"
folder.\n"
				+ "\n*** Please ensure to enter the correct
filename while performing file operations. ***\n\n";
		System.out.println(companyInfo);

		System.out.println(appFunction);
	}
```

Output:

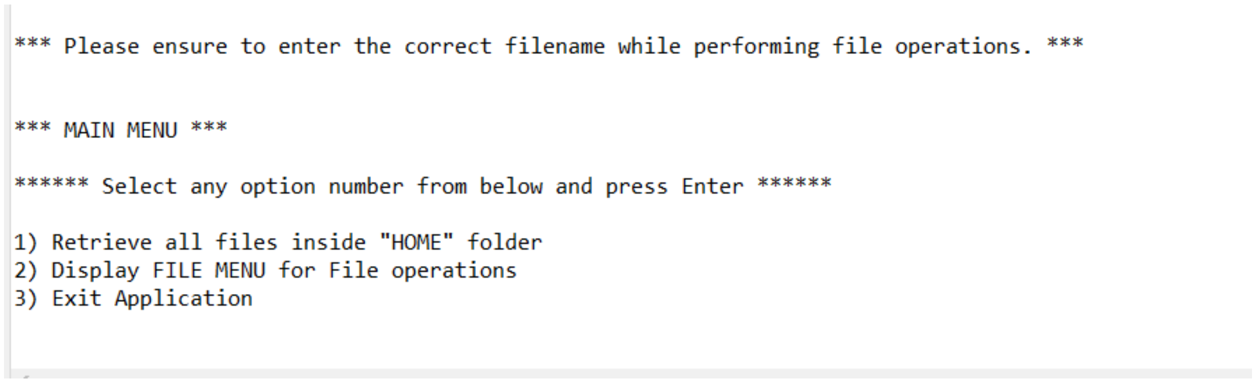


```
*****************************************************

----              WELCOME to LockedMe.com.              ----

---- This application was developed by KEERTHANA BANDARI. ----

*****************************************************

You can use this application to :-
• Retrieve all the file names in the "HOME" folder.
• Add, Delete or Search files in "HOME" folder.

*** Please ensure to enter the correct filename while performing file operations. ***
```

Step 3.2: Writing method to display Main Menu:

```
public static void displayMainMenu() {
		String mainmenu = "*** MAIN MENU ***\n\n" + "****** Select
any option number from below and press Enter ******\n\n"
				+ "1) Retrieve all files inside \"HOME\"
folder\n" + "2) Display FILE MENU for File operations\n"
				+ "3) Exit Application\n";
		System.out.println(mainmenu);

	}
```

Output:

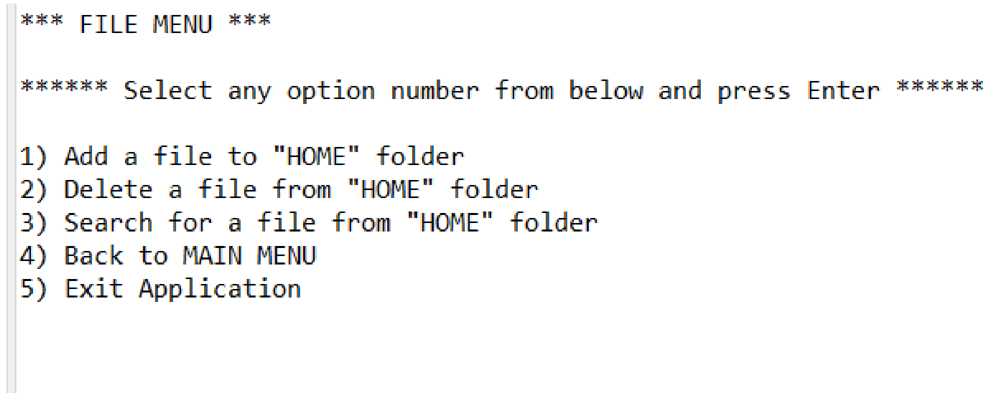


```
*** Please ensure to enter the correct filename while performing file operations. ***


*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application
```

Step 3.3: Writing method to display File Menu for File Operations:

```
public static void displayFileMenu() {
		String fileMenu = "*** FILE MENU ***\n\n" + "****** Select any option number from below and press Enter ******\n\n"
				+ "1) Add a file to \"HOME\" folder\n" + "2) Delete a file from \"HOME\" folder\n"
				+ "3) Search for a file from \"HOME\" folder\n" + "4) Back to MAIN MENU\n" + "5) Exit Application\n";

		System.out.println(fileMenu);
	}
```

Output:



```
*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application
```

Step 4: Writing a program in Java to handle Menu options selected by user (**HandleOptions.java**):

- Select your project and go to File -> New -> Class.
- Enter **HandleOptions** in class name and click on “Finish.”
- **HandleOptions** consists methods for -:

  4.1.Handling input selected by user in Main menu.

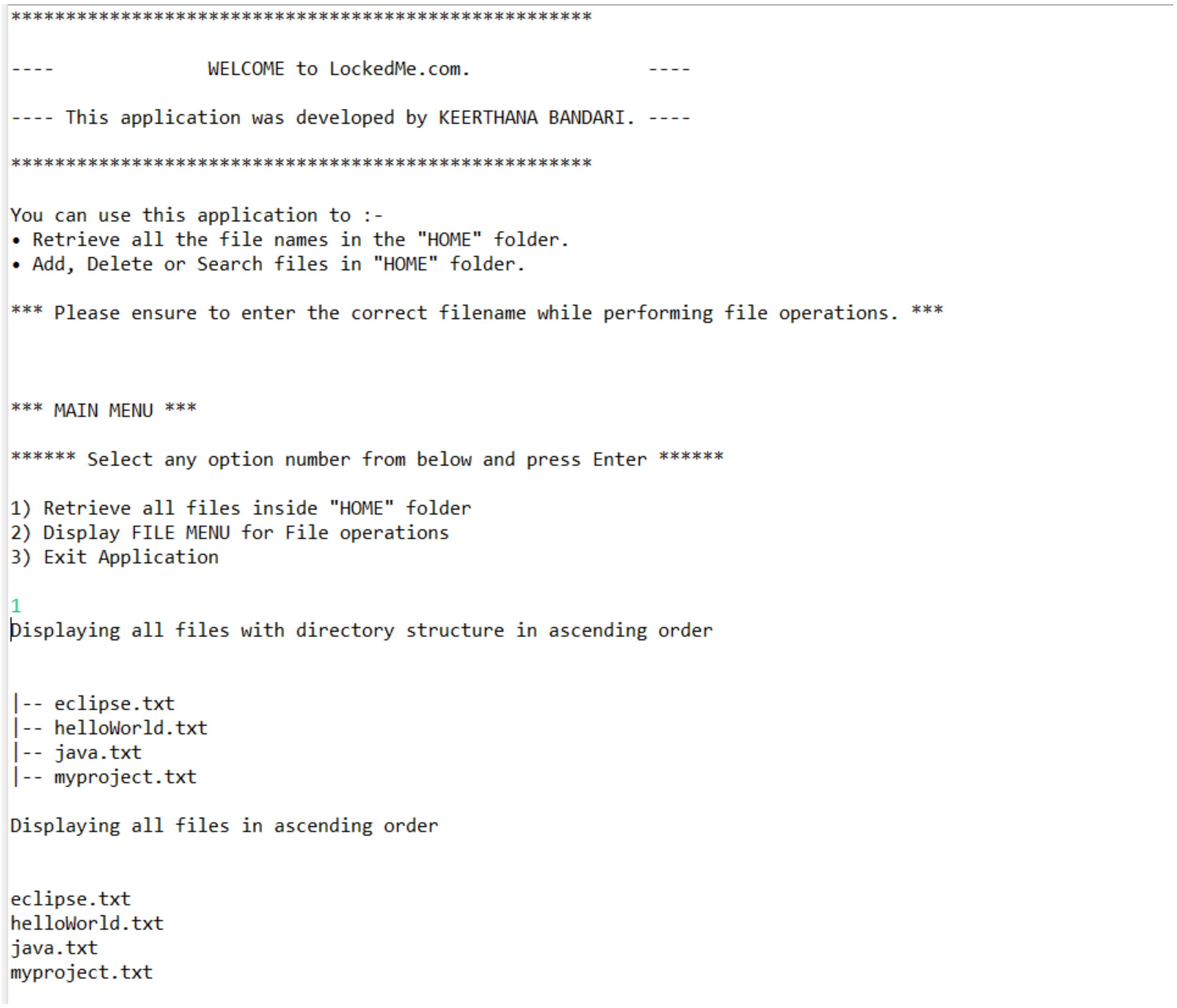
  4.2.Handling input selected by user in File menu for File Operations

Step 4.1: Writing method to handle user input in Main Menu:

```
public static void handleWelcomeScreen() {
		boolean running = true;
		Scanner sc = new Scanner(System.in);
		do {
			try {
				MenuOptions.displayMainMenu();
				int option = sc.nextInt();

				switch (option) {
				case 1:
					FileOperations.displayAllFiles("HOME");
					break;
				case 2:
					HandleOptions.handleFileMenuOptions();
					break;
				case 3:
					System.out.println("Program exited 
successfully.");
					running = false;
					sc.close();
					System.exit(0);
					break;
				default:
					System.out.println("Please select a valid 
option from above.");
				}
			} catch (Exception e) {
				//System.out.println(e.getClass().getName());
```

```
                        System.out.println("Please select a valid option
from above.");

                        handleWelcomeScreen();
                  }
            } while (running == true);
      }
```

Output:



```
*****************************************************

----              WELCOME to LockedMe.com.                ----

---- This application was developed by KEERTHANA BANDARI. ----

*****************************************************

You can use this application to :-
• Retrieve all the file names in the "HOME" folder.
• Add, Delete or Search files in "HOME" folder.

*** Please ensure to enter the correct filename while performing file operations. ***



*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

1
Displaying all files with directory structure in ascending order


|-- eclipse.txt
|-- helloWorld.txt
|-- java.txt
|-- myproject.txt

Displaying all files in ascending order


eclipse.txt
helloWorld.txt
java.txt
myproject.txt
```

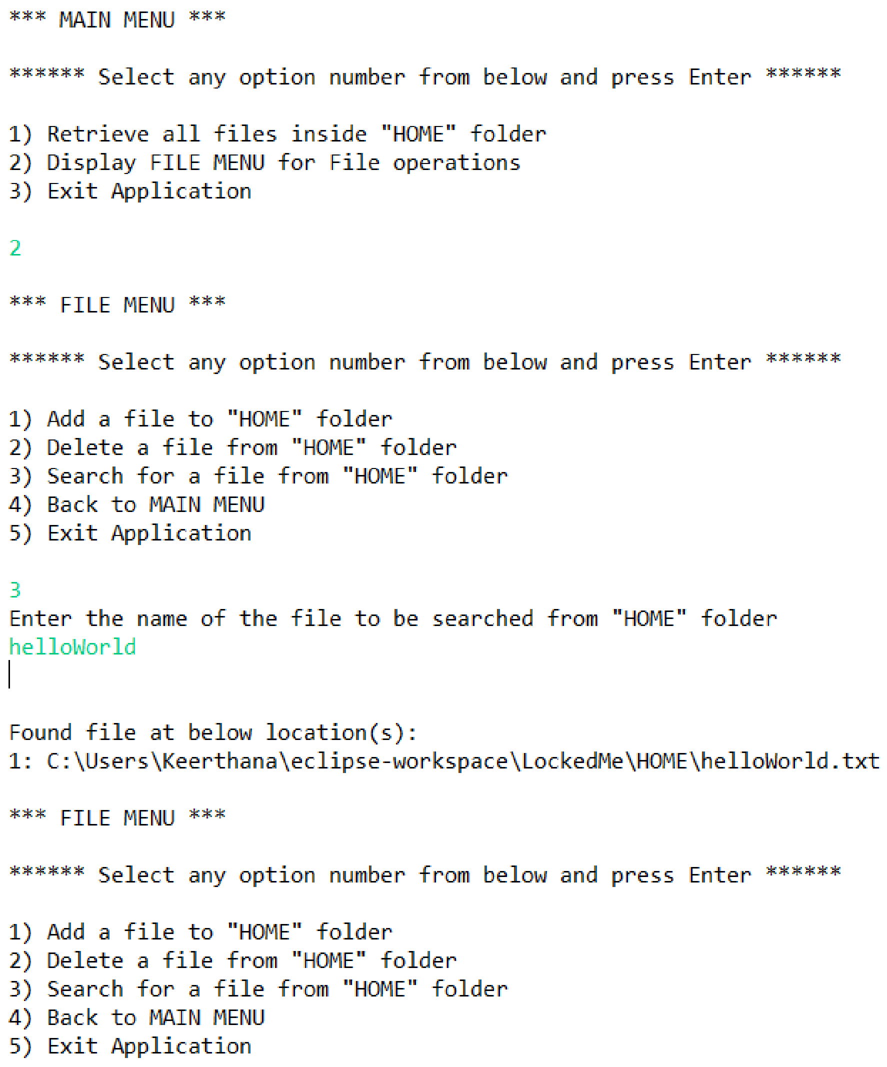
Step 4.2: Writing method to handle user input in File Menu for File Operations:

```java
public static void handleFileMenuOptions() {
		boolean running = true;
		Scanner sc = new Scanner(System.in);
		do {
			try {
				MenuOptions.displayFileMenu();
				FileOperations.createMainFolder("HOME");

				int input = sc.nextInt();
				switch (input) {
```

```
                    case 1:
                        // File Add
                        System.out.println("Enter the name of the
file to be added to the \"HOME\" folder");
                        String fileToAdd = sc.next();

                        FileOperations.createFile(fileToAdd, sc);

                        break;

                    case 2:
                        // File/Folder delete
                        System.out.println("Enter the name of the
file to be deleted from \"HOME\" folder");
                        String fileToDelete = sc.next();

                        FileOperations.createMainFolder("HOME");
                        List<String> filesToDelete =
FileOperations.displayFileLocations(fileToDelete, "HOME");

                        String deletionPrompt = "\nSelect index of
which file to delete?"
                                + "\n(Enter 0 if you want to
delete the file)";
                        System.out.println(deletionPrompt);

                        int idx = sc.nextInt();

                        if (idx != 0) {
    FileOperations.deleteFileRecursively(filesToDelete.get(idx - 1));
                        } else {

                            // If idx == 0, delete all files
displayed for the name
                            for (String path : filesToDelete) {
    FileOperations.deleteFileRecursively(path);
                            }
                        }


                        break;

                    case 3:
                        // File/Folder Search
```

```
                                System.out.println("Enter the name of the
file to be searched from \"HOME\" folder");
                                String fileName = sc.next();

                                FileOperations.createMainFolder("HOME");

      FileOperations.displayFileLocations(fileName, "HOME");


                                break;

                          case 4:
                                // Go to Previous menu
                                return;
                          case 5:

                                // Exit
                                System.out.println("Program exited
successfully.");

                                running = false;
                                sc.close();
                                System.exit(0);

                          default:
                                System.out.println("Please select a valid
option from above.");
                          }
                    } catch (Exception e) {
                          //System.out.println(e.getClass().getName());
                          System.out.println("Please select a valid option
from above.");

                          handleFileMenuOptions();
                    }
              } while (running == true);
      }
}
```

Output:



```
*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

2

*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application

3
Enter the name of the file to be searched from "HOME" folder
helloWorld


Found file at below location(s):
1: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\helloWorld.txt

*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application
```

Step 5: Writing a program in Java to perform the File operations as specified by user (**FileOperations.java**):

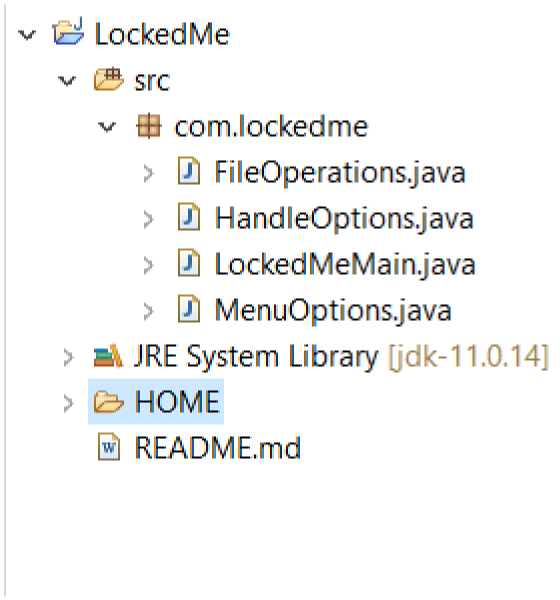
- Select your project and go to File -> New -> Class.
- Enter **FileOperations** in class name and click on "Finish."
- **FileOperations** consists methods for -:

5.1. Creating "HOME" folder in project if it's not already present
5.2. Displaying all files in "HOME" folder in ascending order and also with directory structure.
5.3. Creating a file/folder as specified by user input.
5.4. Search files as specified by user input in "HOME" folder and its subfolders.
5.5. Deleting a file/folder from "HOME" folder

Step 5.1: Writing method to create "HOME" folder in project if it's not present:

```java
public class FileOperations {

	public static void createMainFolder(String folderName) {
		File file = new File(folderName);

		// If file doesn't exist, create the HOME folder
		if (!file.exists()) {
			file.mkdirs();
		}
	}
```

Output:

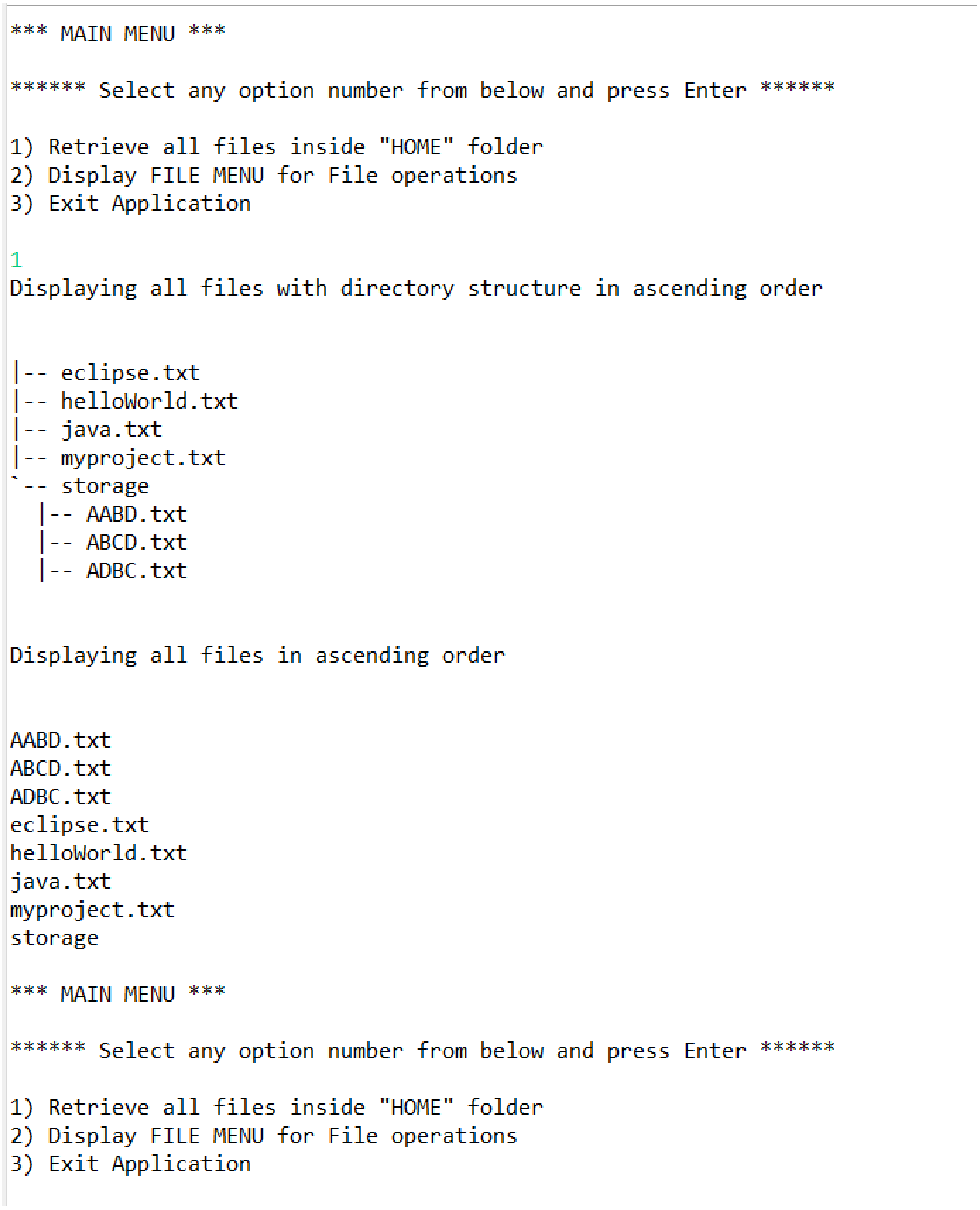


LockedMe
src
com.lockedme
FileOperations.java
HandleOptions.java
LockedMeMain.java
MenuOptions.java
JRE System Library [jdk-11.0.14]
HOME
README.md

Step 5.2: Writing method to display all files in “HOME” folder in ascending order and also with directory structure. (“`--" represents a directory. “|--” represents a file).

```
public static void displayAllFiles(String path) {
		FileOperations.createMainFolder("HOME");
		// All required files and folders inside "HOME" folder 
relative to current
		// folder
		System.out.println("Displaying all files with directory 
structure in ascending order\n\n");

		// listFilesInDirectory displays files along with folder 
structure
		List<String> filesListNames = 
FileOperations.listFilesInDirectory(path, 0, new ArrayList<String>());

		System.out.println("Displaying all files in ascending 
order\n\n");
		Collections.sort(filesListNames);

		filesListNames.stream().forEach(System.out::println);
	}

	public static List<String> listFilesInDirectory(String path, int 
indentationCount, List<String> fileListNames) {
		File dir = new File(path);
		File[] files = dir.listFiles();
		List<File> filesList = Arrays.asList(files);

		Collections.sort(filesList);

		if (files != null && files.length > 0) {
			for (File file : filesList) {

				System.out.print(" ".repeat(indentationCount *
2));

				if (file.isDirectory()) {
					System.out.println("`-- " +
file.getName());

					// Recursively indent and display the files
					fileListNames.add(file.getName());

	listFilesInDirectory(file.getAbsolutePath(), indentationCount +
1, fileListNames);
				} else {
					System.out.println("|-- " +
file.getName());
					fileListNames.add(file.getName());
				}
			}
		} else {
			System.out.print(" ".repeat(indentationCount * 2));
			System.out.println("|-- Empty Directory");
		}
		System.out.println();
		return fileListNames;
	}
```

Output:



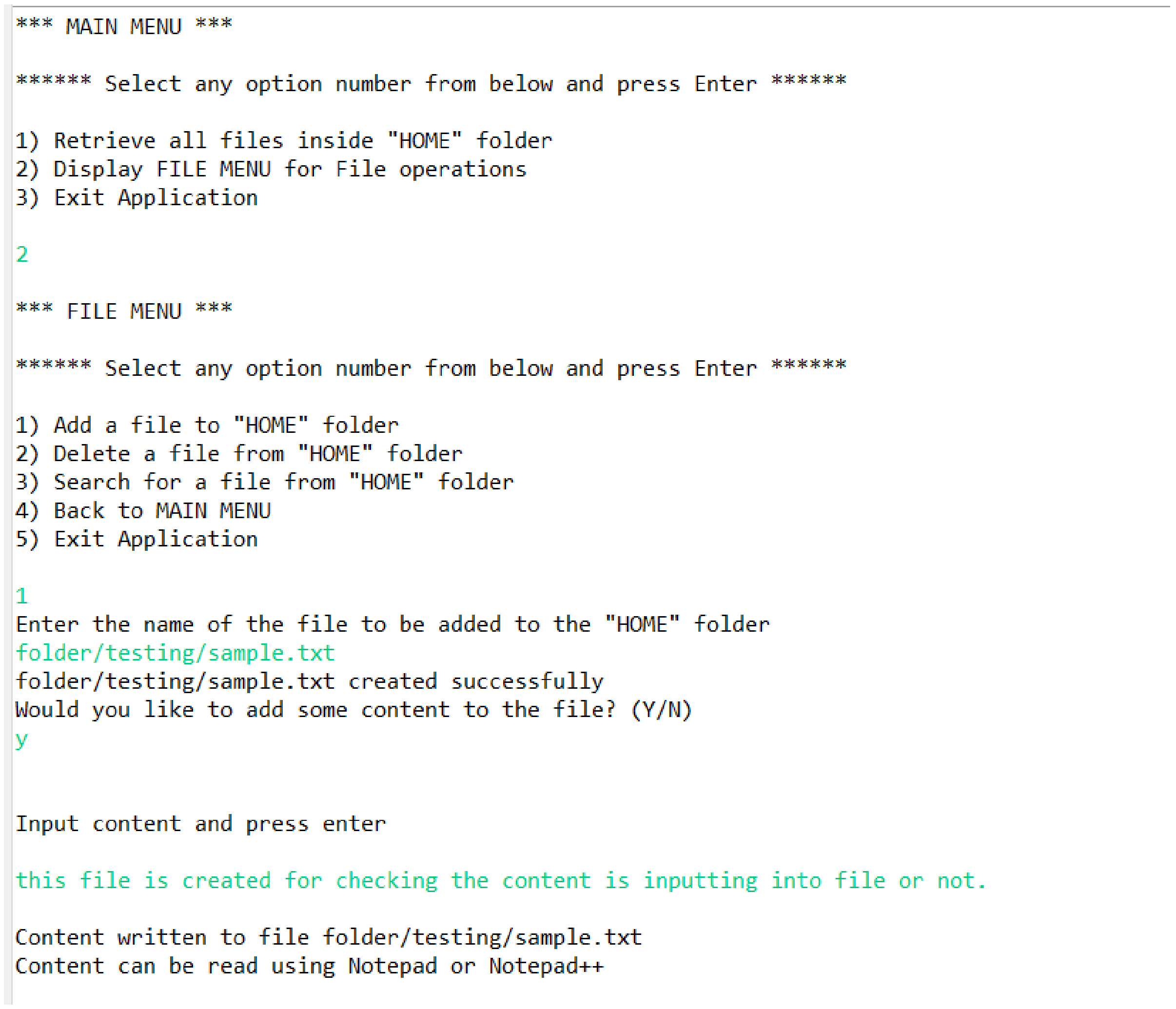
```
*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

1
Displaying all files with directory structure in ascending order


|-- eclipse.txt
|-- helloWorld.txt
|-- java.txt
|-- myproject.txt
`-- storage
  |-- AABD.txt
  |-- ABCD.txt
  |-- ADBC.txt


Displaying all files in ascending order


AABD.txt
ABCD.txt
ADBC.txt
eclipse.txt
helloWorld.txt
java.txt
myproject.txt
storage

*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application
```

Step 5.3: Writing method to create a file/folder as specified by user input:

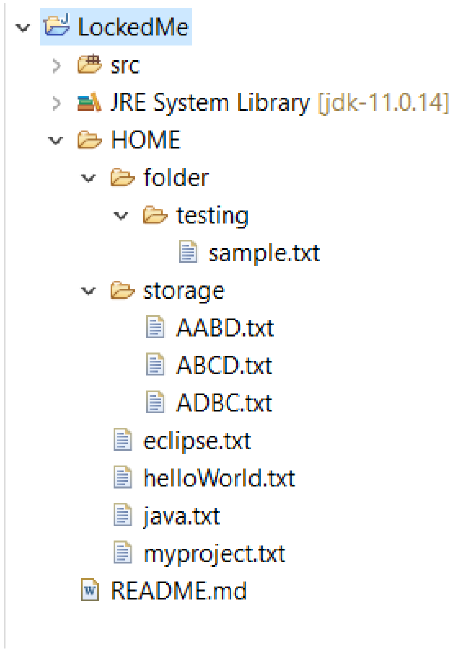
```
public static void createFile(String fileToAdd, Scanner sc) {
		FileOperations.createMainFolder("HOME");
		Path pathToFile = Paths.get("./HOME/" + fileToAdd);
		try {
			Files.createDirectories(pathToFile.getParent());
			Files.createFile(pathToFile);
			System.out.println(fileToAdd + " created 
successfully");

			System.out.println("Would you like to add some content 
to the file? (Y/N)");
			String choice = sc.next().toLowerCase();

			sc.nextLine();
			if (choice.equals("y")) {
				System.out.println("\n\nInput content and press 
enter\n");
				String content = sc.nextLine();
				Files.write(pathToFile, content.getBytes());
				System.out.println("\nContent written to file " + 
fileToAdd);
				System.out.println("Content can be read using 
Notepad or Notepad++");
			}

		} catch (IOException e) {
			System.out.println("Failed to create file " + 
fileToAdd);
			//System.out.println(e.getClass().getName());
		}
	}
```

Output:

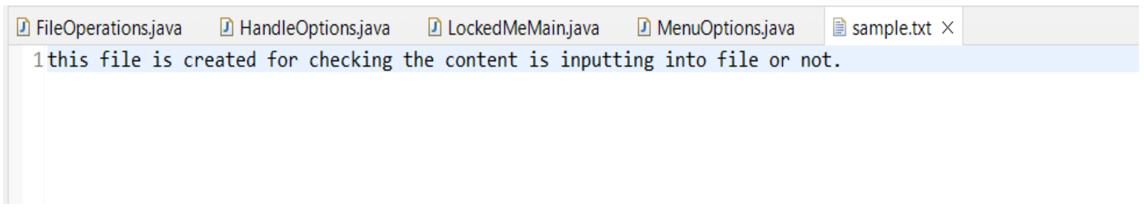
Folders are automatically created along with file



```
*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

2

*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application

1
Enter the name of the file to be added to the "HOME" folder
folder/testing/sample.txt
folder/testing/sample.txt created successfully
Would you like to add some content to the file? (Y/N)
y


Input content and press enter

this file is created for checking the content is inputting into file or not.

Content written to file folder/testing/sample.txt
Content can be read using Notepad or Notepad++
```



```
LockedMe
  src
  JRE System Library [jdk-11.0.14]
  HOME
    folder
      testing
        sample.txt
    storage
      AABD.txt
      ABCD.txt
      ADBC.txt
    eclipse.txt
    helloWorld.txt
    java.txt
    myproject.txt
  README.md
```



```
this file is created for checking the content is inputting into file or not.
```

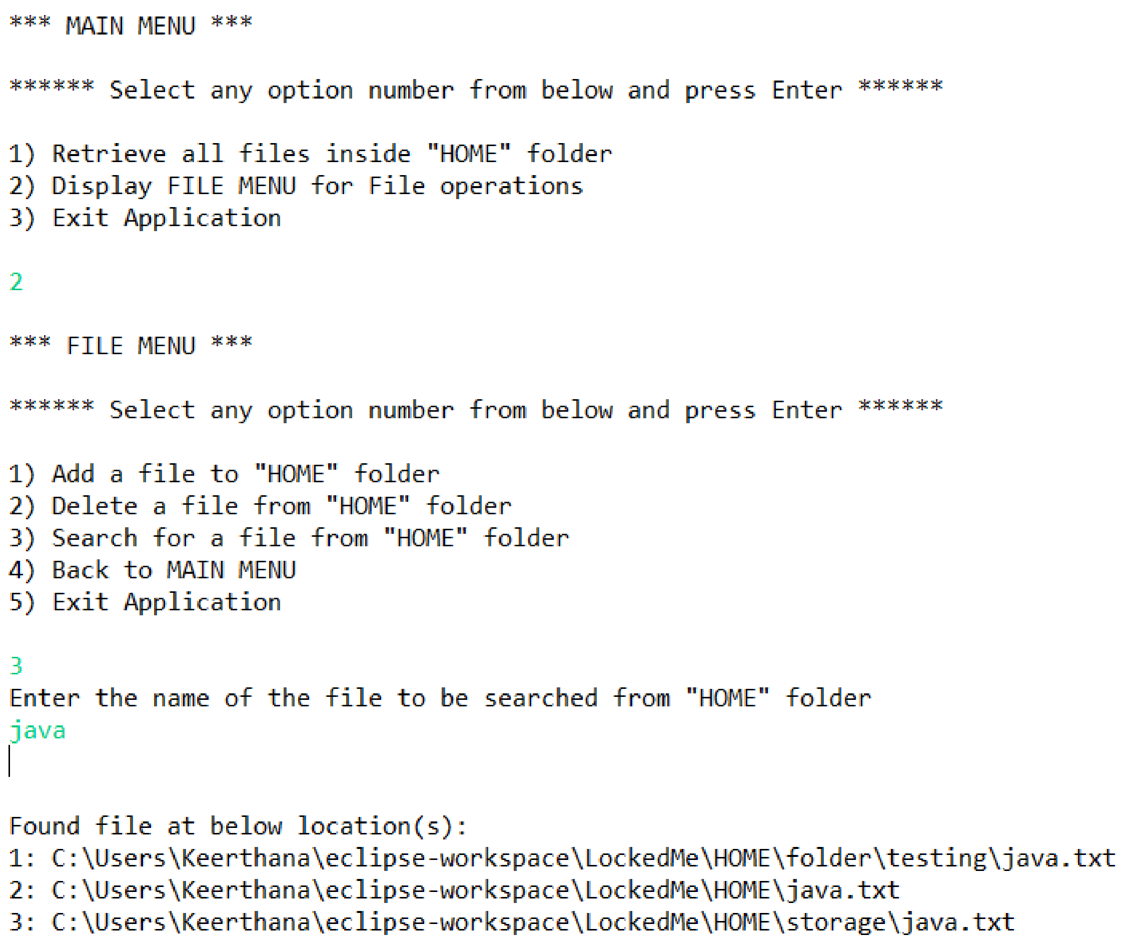
Step 5.4: Writing method to search for all files as specified by user input in “HOME” folder and it’s subfolders:

```
public static List<String> displayFileLocations(String fileName,
String path) {
		List<String> fileListNames = new ArrayList<>();
		FileOperations.searchFileRecursively(path, fileName,
fileListNames);

		if (fileListNames.isEmpty()) {
			System.out.println("\n\n***** Couldn't find any file
with given file name \"" + fileName + "\" *****\n\n");
		} else {
			System.out.println("\n\nFound file at below
location(s):");

			List<String> files = IntStream.range(0,
fileListNames.size())
					.mapToObj(index -> (index + 1) + ": " +
fileListNames.get(index)).collect(Collectors.toList());

			files.forEach(System.out::println);
		}

		return fileListNames;
	}

	public static void searchFileRecursively(String path, String
fileName, List<String> fileListNames) {
		File dir = new File(path);
		File[] files = dir.listFiles();
		List<File> filesList = Arrays.asList(files);

		if (files != null && files.length > 0) {
```

```
                    for (File file : filesList) {

                        if (file.getName().startsWith(fileName)) {
                            fileListNames.add(file.getAbsolutePath());
                        }

                        // Need to search in directories separately to
ensure all files of required
                        // fileName are searched
                        if (file.isDirectory()) {

    searchFileRecursively(file.getAbsolutePath(), fileName,
fileListNames);
                        }
                    }
                }
            }
```

<u>Output:</u>

All files starting with the user input are displayed along with index.



```
*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

2

*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application

3
Enter the name of the file to be searched from "HOME" folder
java


Found file at below location(s):
1: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\folder\testing\java.txt
2: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\java.txt
3: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\storage\java.txt
```

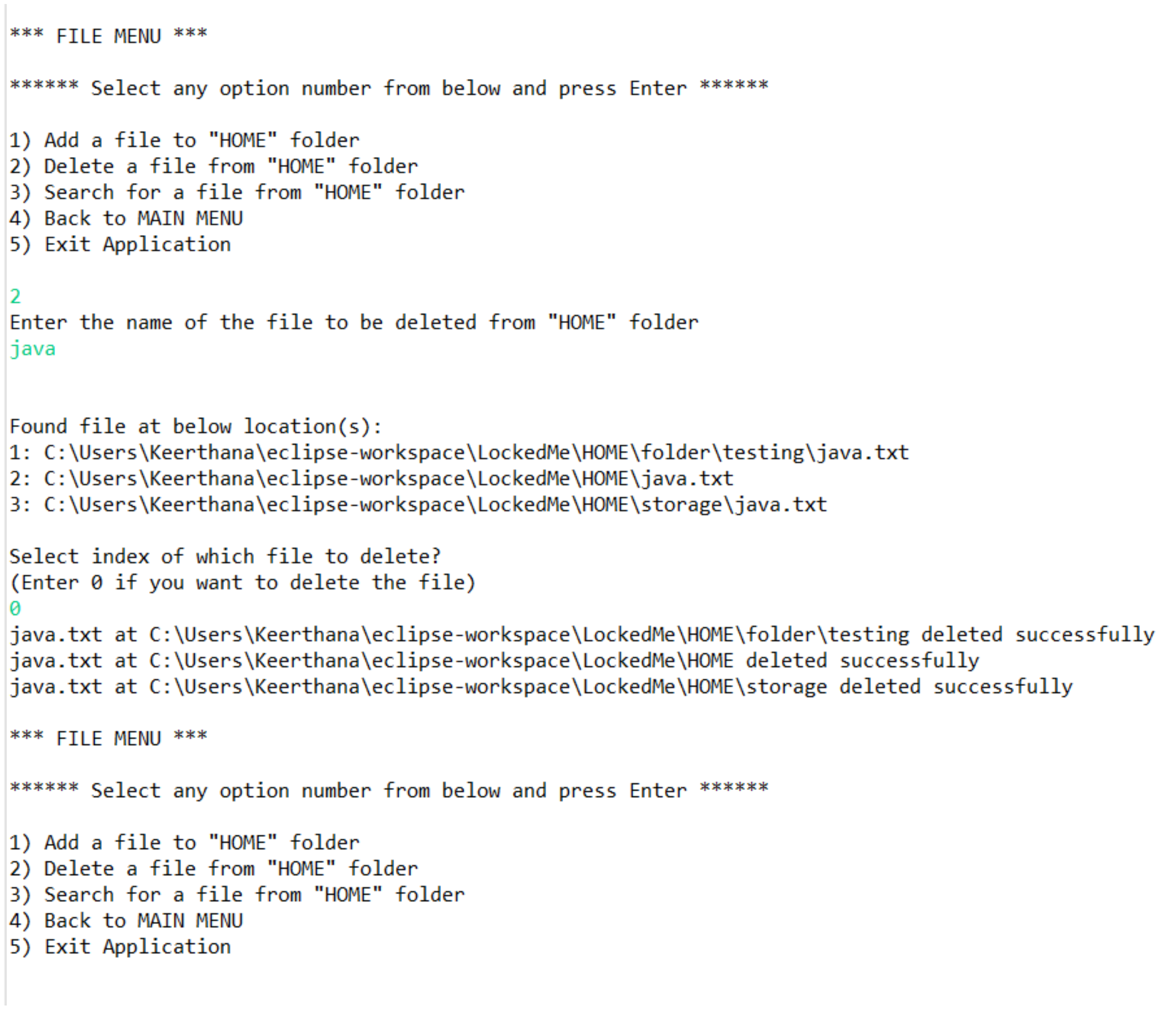
Step 5.5: Writing method to delete file/folder specified by user input in “HOME” folder and its subfolders. It uses the searchFilesRecursively method and prompts user to specify which index to delete. If folder selected, all it’s child files and folder will be deleted recursively. If user wants to delete all the files specified after the search, they can input value 0:

```
public static void deleteFileRecursively(String path) {

        File currFile = new File(path);
        File[] files = currFile.listFiles();
```

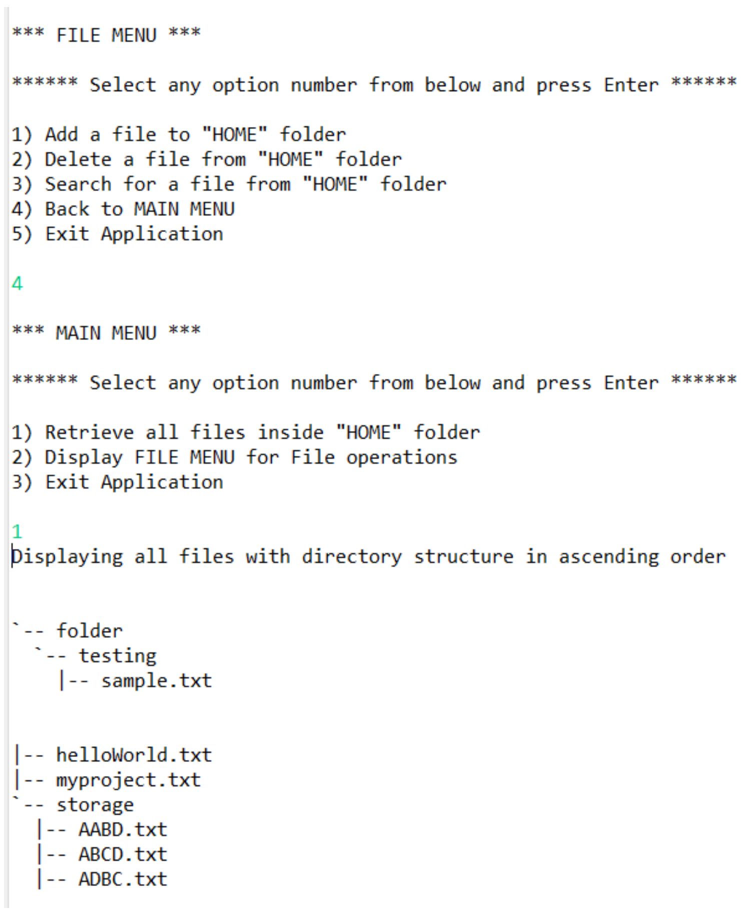
```
            if (files != null && files.length > 0) {
                  for (File file : files) {

                        String fileName = file.getName() + " at " +
file.getParent();
                        if (file.isDirectory()) {

      deleteFileRecursively(file.getAbsolutePath());
                        }

                        if (file.delete()) {
                              System.out.println(fileName + " deleted
successfully");
                        } else {
                              System.out.println("Failed to delete " +
fileName);
                        }
                  }
            }

            String currFileName = currFile.getName() + " at " +
currFile.getParent();
            if (currFile.delete()) {
                  System.out.println(currFileName + " deleted
successfully");
            } else {
                  System.out.println("Failed to delete " +
currFileName);
            }
      }
}
```

Output:

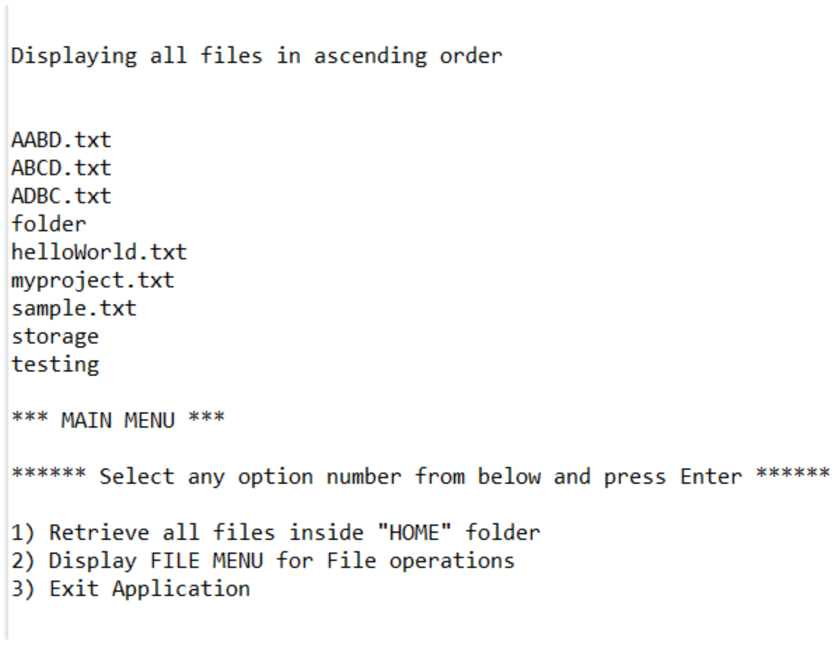
To verify if file is deleted on Eclipse, right click on Project and click “Refresh”.



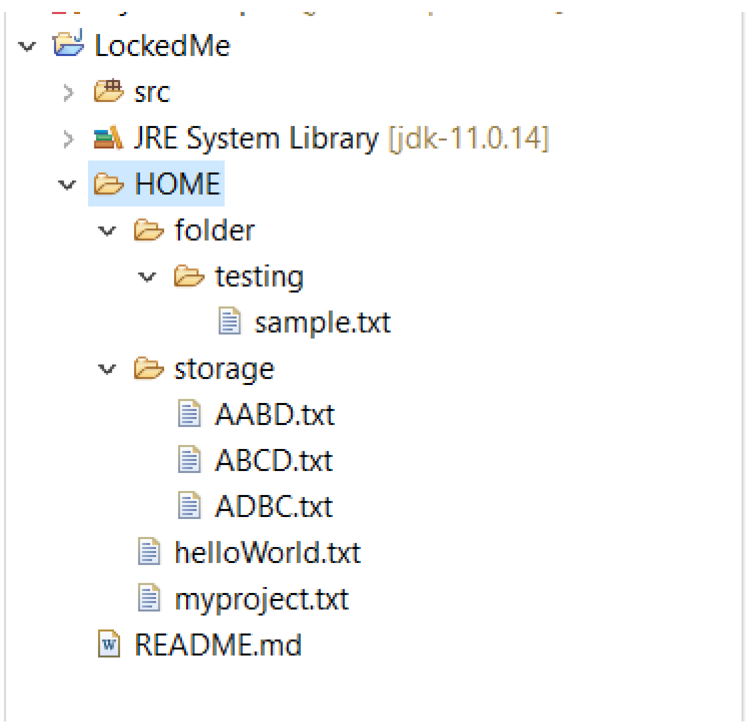
```
*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application

2
Enter the name of the file to be deleted from "HOME" folder
java


Found file at below location(s):
1: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\folder\testing\java.txt
2: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\java.txt
3: C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\storage\java.txt

Select index of which file to delete?
(Enter 0 if you want to delete the file)
0
java.txt at C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\folder\testing deleted successfully
java.txt at C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME deleted successfully
java.txt at C:\Users\Keerthana\eclipse-workspace\LockedMe\HOME\storage deleted successfully

*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application
```



```
*** FILE MENU ***

****** Select any option number from below and press Enter ******

1) Add a file to "HOME" folder
2) Delete a file from "HOME" folder
3) Search for a file from "HOME" folder
4) Back to MAIN MENU
5) Exit Application

4

*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application

1
Displaying all files with directory structure in ascending order


`-- folder
  `-- testing
    |-- sample.txt


|-- helloWorld.txt
|-- myproject.txt
`-- storage
  |-- AABD.txt
  |-- ABCD.txt
  |-- ADBC.txt
```



```
Displaying all files in ascending order


AABD.txt
ABCD.txt
ADBC.txt
folder
helloWorld.txt
myproject.txt
sample.txt
storage
testing

*** MAIN MENU ***

****** Select any option number from below and press Enter ******

1) Retrieve all files inside "HOME" folder
2) Display FILE MENU for File operations
3) Exit Application
```



<u>**Step 6:** Pushing the code to GitHub repository:</u>

- Open your command prompt and navigate to the folder where you have created your files.

  <u>Command</u>: cd <folder path>

- Initialize repository using the following command:

  <u>Command: </u>git init

- Add all the files to your git repository using the following command:

  <u>Command</u>: git add .

- Commit the changes using the following command:

  <u>Command:</u> git commit -m <commit message>

- Push the files to the folder you initially created using the following command:

  Command: **git push -u origin master**

  Github: https://github.com/keerthana-777/LockedMe.git

Unique Selling Points of the Application:

1. The application can take any file/folder name as input. Even if the user wants to create nested folder structure, user can specify the relative path, and the application takes care of creating the required folder structure.

2. The application is designed to keep on running and taking user inputs even after exceptions occur. To terminate the application, appropriate option needs to be selected.

3. User is also provided the option to write content if they want into the newly created file.

4. User is also provided the option to write content if they want into the newly created file.

5. The application doesn't restrict user to specify the exact filename to search/delete file/folder. They can specify the starting input, and the program searches all files/folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

6. The application also allows user to delete folders which are not empty.

7. The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

8. When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.

   8.1. Ascending order of folders first which have files sorted in them,
   8.2. Ascending order of all files and folders inside the “HOME” folder.

9. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code. Application has been developed keeping in mind that there should be very less “hardcoding” of data.

Conclusions:

Further enhancements to the application can be made which may include:

- Asking user to verify if they really want to delete the selected directory if it’s not empty.
- Conditions to check if user is allowed to delete the file or add the file at the specific locations.
- Retrieving files/folders by different criteria like Type, Last Modified, Date, etc.
- Allowing user to append data to the file.
- Modify the file as per the users need.