LockedMe – Virtual Key for Repositories

This document contains sections for:

The code for this project is hosted at https://github.com/vaishraw/practiceprojects

The project is developed by Vaishnavi Rawat.

Sprints planning and Task completion

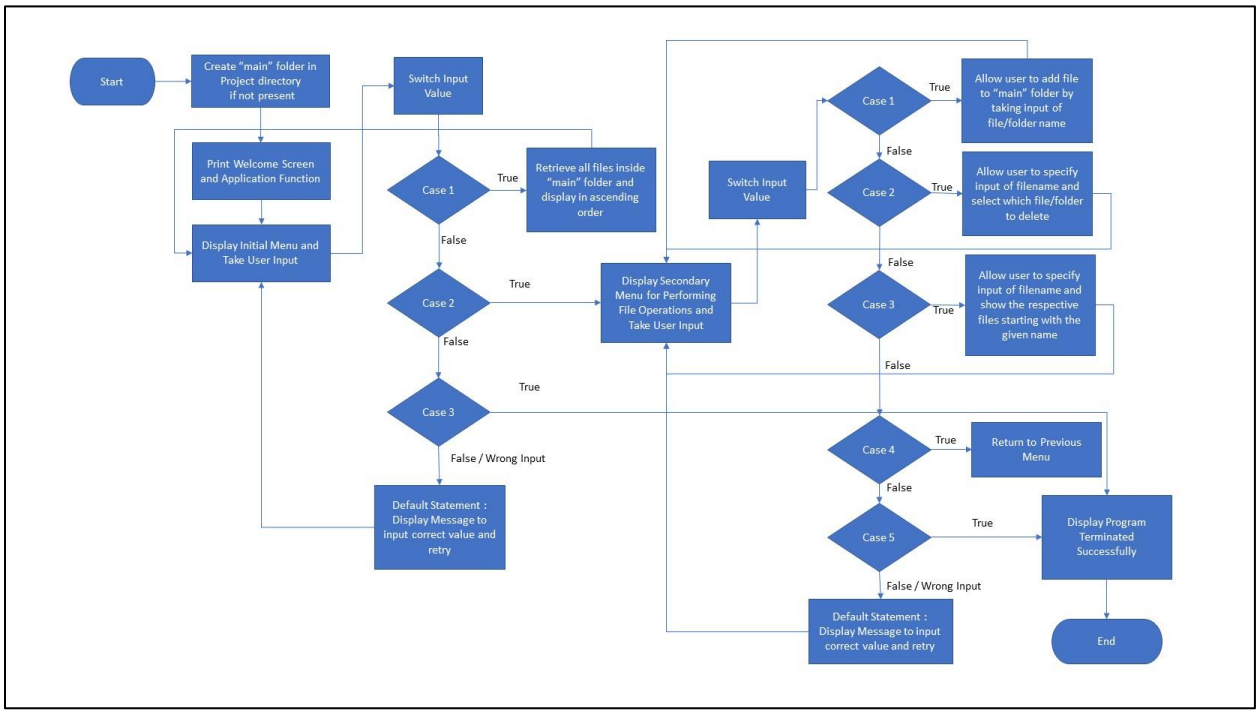
The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

- Creating the flow of the application
- Initializing git repository to track changes as development progresses.
- Writing the Java program to fulfill the requirements of the project.
- Testing the Java program with different kinds of User input
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Core concepts used in project

Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Streams API

Flow of the Application



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. Creating the project in Eclipse
2. Writing a program in Java for the entry point of the application (**Main.java**)
3. Writing a program in Java to display Menu options available for the user (**Menu.java**)
4. Writing a program in Java to handle Menu options selected by user (**operations.java**)
5. 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 **Main** 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 **(Main.java)**

```
public class Main {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Menu.printWlcmScreen("LockedMe.com", "Vaishnavi Rawat");
		operations.optionSelection();


	}

}
```

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

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

3.1. Displaying Welcome Screen

3.2. Displaying Main Menu

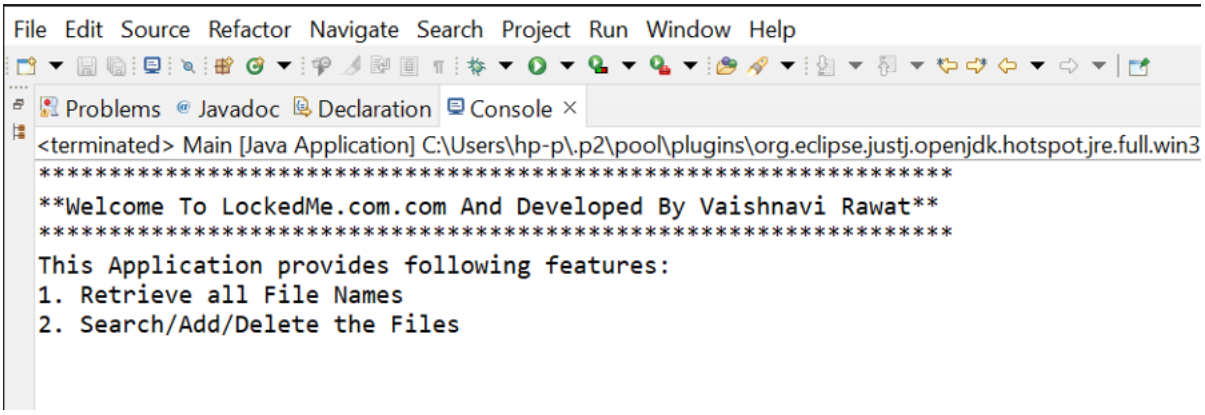
3.3. Displaying File Menu for File Operations available

Step 3.1: Writing method to display Welcome Screen

```
public class Menu {

	public static void printWlcmScreen(String AppName, String DevName) {
		String compinfo =
String.format("\n**********************************************************************"
				+ "\n**Welcome To %s.com And Developed By %s**" +
"\n**********************************************************************", AppName,
```

```
                        DevName);
                String appfun = ("This Application provides following features:\n" + "1.
Retrieve all File Names\n"
                        + "2. Search/Add/Delete the Files\n");
                System.out.println(compinfo);
                System.out.println(appfun);
        }
```



Step 3.2: Writing method to display Main Menu

```
public static void DisplayMainMenu() {
                String mainmenu = ("\n\n****Welcome To Main Menu****\n" + "1. Retrieve
All Files in Ascending Order\n"
                        + "2. Display File Operations Menu\n" + "3. Exit the
Application\n");
                System.out.println(mainmenu);

        }
```

```
****Welcome To Main Menu****
1. Retrieve All Files in Ascending Order
2. Display File Operations Menu
3. Exit the Application
```

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

```
public static void DisplayFileMenu() {
                String fmenu = ("\n\n***Welcome To File Menu Operations****\n" + "1. Add
a New File\n"
                        + "2. Delete the Existing File\n" + "3. Search the File\n"
+ "4. Show Previous Menu\n"
                        + "5.Exit the Application\n");
                System.out.println(fmenu);
        }
}
```

```
***Welcome To File Menu Operations****
1. Add a New File
2. Delete the Existing File
3. Search the File
4. Show Previous Menu
5.Exit the Application
```

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

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

4.1. Handling input selected by user in initial Menu

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

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

```
import java.util.Scanner;
import java.util.Set;
import java.util.TreeSet;

public class operations {
	static Set<String> obj2;
	static Set<String> obj;

	public static void optionSelection() {
		obj2 = new TreeSet<String>();
		obj = new TreeSet<String>();
		obj.add("abc.txt");
		obj.add("xyz.txt");
		obj.add("pqr.docx");
		obj2.add("def.pdf");
		obj2.add("tuv.pdf");
		obj.addAll(obj2);
		Scanner sc = new Scanner(System.in);
		int ch;
		do {
			Menu.DisplayMainMenu();
			System.out.println("Enter your choice.");
			ch = sc.nextInt();
			switch (ch) {
			case 1:
				System.out.println("All Files in the folder: ");
				System.out.println(obj);
				break;
			case 2:
```

```
                        System.out.println("File Operations Menu: ");
                        operations.FileMenuOperations();
                        break;
                case 3:
                        System.out.println("Application Exited Successfully.");
                        sc.close();
                        break;
                default:
                        System.out.println("Enter Valid Choice.");

                }
        } while (ch != 3);

    }
```

```
**********************************************************************
**Welcome To LockedMe.com.com And Developed By Vaishnavi Rawat**
**********************************************************************
This Application provides following features:
1. Retrieve all File Names
2. Search/Add/Delete the Files



****Welcome To Main Menu****
1. Retrieve All Files in Ascending Order
2. Display File Operations Menu
3. Exit the Application

Enter your choice.
1
All Files in the folder:
[abc.txt, def.pdf, pqr.docx, tuv.pdf, xyz.txt]


****Welcome To Main Menu****
1. Retrieve All Files in Ascending Order
2. Display File Operations Menu
3. Exit the Application

Enter your choice.
3
Application Exited Successfully.
```

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

```
private static void FileMenuOperations() {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		int ch;
		do {
			Menu.DisplayFileMenu();
			System.out.println("Enter Your Choice.");
			ch = sc.nextInt();
			switch (ch) {
			case 1:
				System.out.println("Enter the name of File to be Added:
");
				String fadd = sc.next();
				obj2.add(fadd);
				obj.addAll(obj2);
				System.out.println("File Added Successfully");
				System.out.println(obj);
				break;
			case 2:
				System.out.println("Enter the name of file you want to
delete: ");
				String fdel = sc.next();
				if (obj.contains(new String(fdel))) {
					obj.remove(new String(fdel));
					System.out.println("File Deleted Successfully");
					System.out.println(obj);
				} else {
					System.out.println("Entered file do not Exist");
				}
				break;
			case 3:
				System.out.println("Enter the Name of File you want to
Search: ");
				String fsearch = sc.next();
				if (obj.contains(new String(fsearch))) {
					System.out.println(fsearch + " file found
Successfully.");
				} else {
					System.out.println(fsearch + " file not found.");
				}
				break;
			case 4:
				System.out.println("Return TO Main Menu:");
				//operations.optionSelection();
				return;

			case 5:
				System.out.println("Application Exited Successfully.");
				sc.close();
				System.exit(0);

			default:
				System.out.println("Enter Valid Choice.");
			}
```

```
            } while (ch != 5);
        }
}
```

```
****Welcome To Main Menu****
1. Retrieve All Files in Ascending Order
2. Display File Operations Menu
3. Exit the Application

Enter your choice.
2
File Operations Menu:


***Welcome To File Menu Operations****
1. Add a New File
2. Delete the Existing File
3. Search the File
4. Show Previous Menu
5.Exit the Application

Enter Your Choice.
3
Enter the Name of File you want to Search:
abc.txt
abc.txt file found Successfully.


***Welcome To File Menu Operations****
1. Add a New File
2. Delete the Existing File
3. Search the File
4. Show Previous Menu
5.Exit the Application

Enter Your Choice.
```

Step 5: Pushing the code to GitHub repository

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

  cd <folder path>

- Initialize repository using the following command:

git init

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

  git add .

- Commit the changes using the following command:

  git commit . -m “changes”

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

  git push -u origin master

Unique Selling Points of the Application

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

2. The application can take any file name as input.

3. The application doesn’t restrict user to specify the exact filename to search/delete file They can specify the starting input, and the program searches all files starting with the value and displays it.

4. 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.

5. 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:

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