VIRTUAL KEY For REPOSITORY

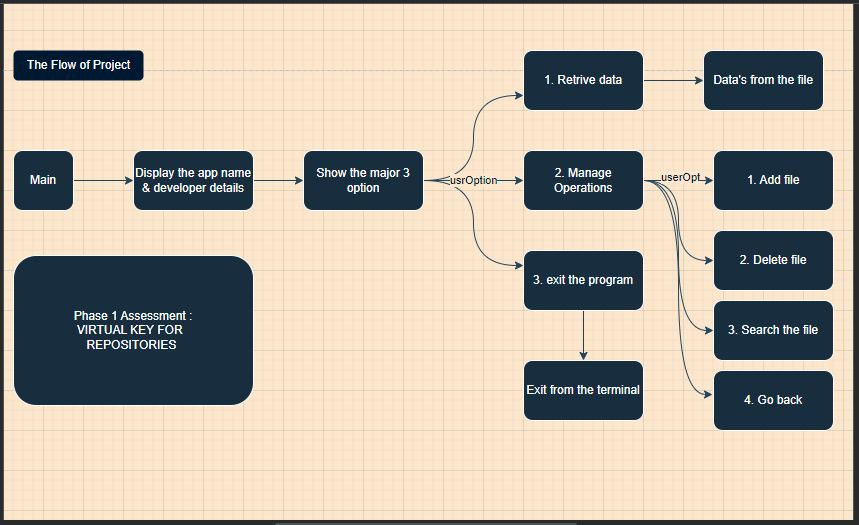
**This document contains the:**

* [Sprint Planning](#_top)
* [Flow of Project](#_top)
* [Concepts used in Project](#_top)
* [Working on Eclipse IDE](#_top)
* [Conclusion](#_top)

**Sprint Planning:**

For this project, I’ve planned to complete the project in 2 Sprints. And the task is completed according to that.

**Flow of Project:**



**Concepts used in Project:**

* File Handling
* Flow Control
* Exeption Handling

**Working on Eclipse IDE:**

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 LockedMain 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 (**LockedMain.java**)

package LockedMe.com;

import java.io.IOException;

public class LockedMain {

public static void main(String[] args) throws IOException {

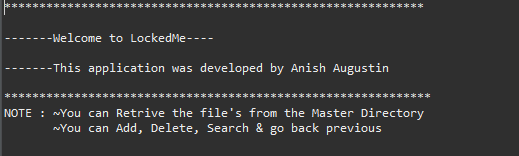
// Calling a method to display the application and the developer details

DisplayRecords.*entryRecords*("Anish Augustin");

//As it shows the application name and developer details, the program will directly start from this function

LockedFile.*beginingOfProgram*();

Output :



}

}

**Step 3:** Writing a new class to display all the options of the application (**DisplayRecords.Java**).

package LockedMe.com;

//This class will be shown in the output

public class DisplayRecords {

public static void entryRecords(String Devep) {

// Creating the application name and developer name

// In the NOTE point, the user get to know what all things can do in this application

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n"

+ "\r\n"

+ "-------Welcome to LockedMe----\r\n"

+ "\r\n"

+ "-------This application was developed by "+ Devep +" \r\n"

+ "\r\n"

+ "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

+ "\r\n"

+ "NOTE : ~You can Retrive the file's from the Master Directory\r\n"

+ " ~You can Add, Delete, Search & go back previous\r\n"

+ "\r\n");

}

// Creating method for the user idea, what to do first

public static void initialOperationRecords() {

//This is is the beginning of the Process

//The user have to click either 1 or 2 or 3 to do the functions

System.***out***.println("\*\*\*\*\*\*\*\*\*\* Select your option and press the ENTER key \*\*\*\*\*\*\*\*\r\n"

+"\r\n"

+ "---->1. Retrieving the file names in an ascending order\r\n"

+ "---->2. Manage Operations\r\n"

+ "---->3. Close the application ");

}

// Creating a method to display the Operations

public static void displayOperationRecords() {

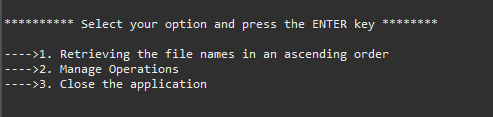
// For user interactions

System.***out***.println("1. Add file in the Master Directory\r\n" + "2. Delete file From the Master Directory\r\n"

+ "3. Search the file in the Master Directory\r\n" + "4. Go back to Previous Option\r\n" + "");

}

Output:



}

**Step 4:** Writing a new class to implement the backend functions (**LockedFile.java**).

The class consist of two Static variables

Specifying each methods:

* public static void fileCreation()
* public static void beginingOfProgram() throws IOException
* private static void manageOperation() throws IOException
* private static void filesinFolder() throws IOException
* private static void addFile() throws IOException
* private static void deleteFile() throws IOException
* private static void searchFile() throws IOException
* private static void programEnds()

**public static void fileCreation():**

public static void fileCreation() {

*Path* = System.*getProperty*("user.dir");

*fileName* = new File(*Path* + "/ Master");

if (!*fileName*.exists()) {

*fileName*.mkdirs();

}

}

**public static void beginingOfProgram() throws IOException:**

// Backend process for user interactions starts here.

public static void beginingOfProgram() throws IOException {

// Calling "filecreation" method to check the directory is created or not

*fileCreation*();

// Calling a method from another class to display the process can done

DisplayRecords.*initialOperationRecords*();

Scanner sc = new Scanner(System.***in***);

int userOpt = sc.nextInt();

// Creating switch case to get the userOption to the process

switch (userOpt) {

case 1:

*filesinFolder*();

break;

case 2:

*manageOperation*();

break;

case 3:

*programEnds*();

break;

default:

System.***out***.println("Invalid option, Please enter the right option");

}

}

**private static void manageOperation() throws IOException:**

// To process the Main operations like add, delete, search & go back

private static void manageOperation() throws IOException {

*fileCreation*();

DisplayRecords.*displayOperationRecords*();

Scanner sc = new Scanner(System.***in***);

int userOpt = sc.nextInt();

switch (userOpt) {

case 1:

*addFile*();

break;

case 2:

*deleteFile*();

break;

case 3:

*searchFile*();

break;

case 4:

*beginingOfProgram*();

break;

default:

System.***out***.println("Invalid Option, Please enter the right one");

}

}

**private static void filesinFolder() throws IOException:**

// To see the files in the directory

private static void filesinFolder() throws IOException {

if (*fileName*.list().length > 0) {

String[] list = *fileName*.list();

Arrays.*sort*(list);

for (String files : list) {

System.***out***.println("--- " + files);

}

System.***out***.println("Files are displayed");

System.***out***.println();

*beginingOfProgram*();

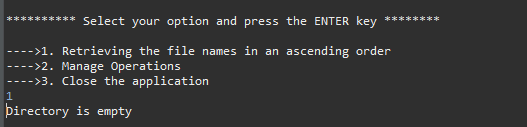
} else {

System.***out***.println("Directory is empty");

System.***out***.println();

*beginingOfProgram*();

Output:



}

}

**private static void addFile() throws IOException:**

// To add files in the directory

private static void addFile() throws IOException {

// **TODO** Auto-generated method stub

System.***out***.println("Enter the file name");

Scanner sc = new Scanner(System.***in***);

String fName = sc.nextLine();

File file = new File(*fileName* + "/" + fName);

if (!file.exists()) {

file.createNewFile();

System.***out***.println("File Created. \r\n" + "Want to write something in the file(Yes/No)?");

String useOpt = sc.nextLine();

if (useOpt.equalsIgnoreCase("Yes")) {

FileWriter Writer = new FileWriter(*fileName* + "/" + fName);

System.***out***.println("\nInput content and press enter\n");

String content = sc.nextLine();

Writer.write(content);

Writer.close();

System.***out***.println("\nContent written to file " + fName);

System.***out***.println("Content can be read using Notepad or Notepad++");

System.***out***.println();

*manageOperation*();

}

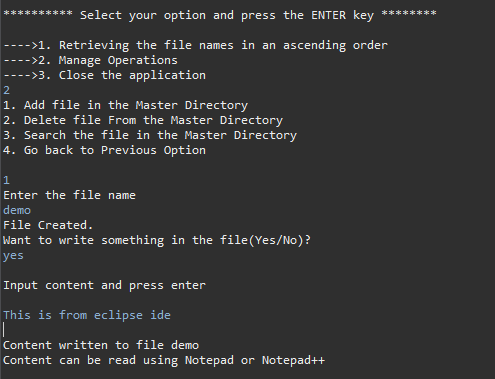
*manageOperation*();

} else {

System.***out***.println("NOTE - !File Name " + fName + " already exist, Enter a new name");

*addFile*();

Output:



**private static void deleteFile() throws IOException:**

private static void deleteFile() throws IOException {

System.***out***

.println("Enter the file name \r\n" + "\r\n" + "?Please ensure that you're entering the Correct name ");

Scanner sc = new Scanner(System.***in***);

String delFile = sc.nextLine();

File file = new File(*fileName* + "/" + delFile);

String[] list = *fileName*.list();

for (String files : list) {

if (delFile.equals(files) && file.delete()) {

System.***out***.println("File " + delFile + " has Successfully deleted");

System.***out***.println();

*manageOperation*();

}

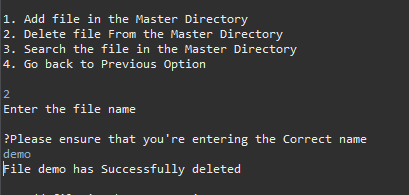
}

System.***out***.println("No file exist in the name ");

System.***out***.println();

*manageOperation*();

Output:



}

**private static void searchFile() throws IOException:**

private static void searchFile() throws IOException {

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter the file name that to be removed/deleted from the directory\r\n" + "\r\n"

+ "?Please ensure that you're entering the Correct name ");

String fileSearch = sc.nextLine();

String[] list = *fileName*.list();

for (String file : list) {

if (fileSearch.equals(file)) {

System.***out***.println(fileSearch + " File Found");

System.***out***.println(fileSearch + " is located in " + *fileName*.getAbsolutePath());

System.***out***.println();

*manageOperation*();

}

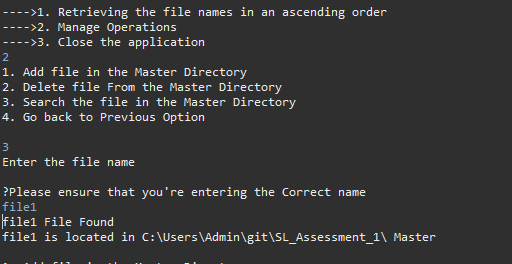
}

System.***out***.println("File Not Found");

System.***out***.println();

*manageOperation*();

Output:



}

**private static void programEnds():**

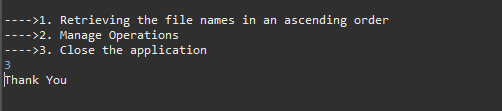
// To end the program

private static void programEnds() {

System.***out***.println("Thank You");

System.*exit*(0);

Output:



C

**Step 5:** Upload files 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 <commit message>
* Push the files to the folder you initially created using the following command:
* git push -u origin master

**Unique Selling Points:**

1. This application is use to create file, add data’s to file, delete file & search for file.
2. This application will keep on running until user close the program.
3. It allows the user to write content on the file created by the user.

**Conclusion:**

Further enhancements to the application can be made which may include

* Accessibility of changing the path of the file and folder
* User should able to delete the repository also
* User should able to create a file inside the file