**Write UP-LockedMe – Virtual Key for Repositories**

This document contains sections for:

* [Sprint planning and Task completion](#Sprint_plan)
* [Core concepts used in project](#Core_concepts)
* [Flow of the Application](#Flow).
* [Demonstrating the product capabilities, appearance, and user interactions.](#Product_capability)
* [Unique Selling Points of the Application](#USP)
* [Conclusions](#Conclusions)

The code for this project is hosted at **https://github.com/prasanthbaskaran25/Java-Fsd.git**

The project is developed by **Prasanth B**.

## **Sprints planning and Task completion**

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

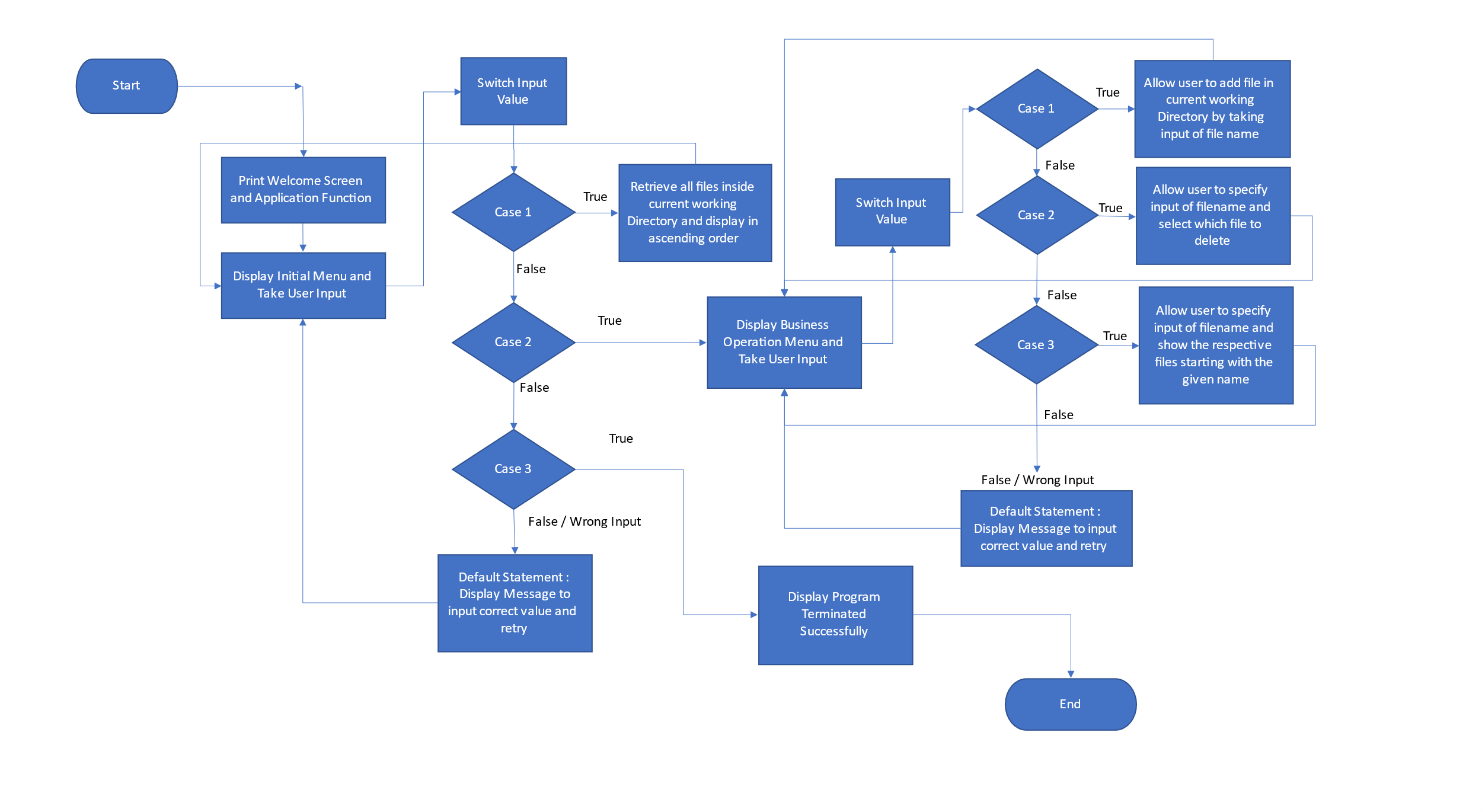
* Creating the flow of the application
* Initializing git repository to track changes as development progresses.
* Writing the Java program to fulfill the requirements of the project.
* Testing the Java program with different kinds of User input
* Pushing code to GitHub.
* Creating this specification 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, beloware the sub-sections configured to highlight appearance and user interactions for the project:

1. [Creating the project in Eclipse](#Step_1)
2. [Writing a program in Java to perform the File operations and entry point as specified by user (**VirtualKeyRepositories.java**)](#Step_5)
3. [Pushing the code to GitHub repository](#Step_6)

## **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 **VirtualKeyRepositories** 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 (**VirtualKeyRepositories.java**)

**package** Demo;

**import** java.util.Scanner;

**import** java.io.File;

**import** java.io.IOException;

**public** **class** VirtualKeyRepositories {

**void** displayfile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

File f=**new** File(path);

//display operation

File filenames[]=f.listFiles();

**for**(File ff:filenames)

{

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

}

}

**void** addfile() **throws** IOException{

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

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

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

//create a new file

**boolean** b=f.createNewFile();

**if**(b!=**true**)

{

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

}

**else**

{

System.***out***.println("File Created");

}

}

**void** deletefile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter a File Name");

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

//delete operation

f.delete();

System.***out***.println("File Gets Deleted");

}

**void** searchfile() {

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter the Filename to Search");

String filename=sc.next();

File f=**new** File(path);

//search operation

**int** flag=0;

File filenames[]=f.listFiles();

**for**(File ff:filenames)

{

**if**(ff.getName().equals(filename))

{

flag=1;

**break**;

}

**else**

{

flag=0;

}

}

**if** (flag==1)

{

System.***out***.println("File is found");

}

**else**

{

System.***out***.println("File is not found");

}

}

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

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

VirtualKeyRepositories dp=**new** VirtualKeyRepositories();

VirtualKeyRepositories p=**new** VirtualKeyRepositories();

VirtualKeyRepositories d=**new** VirtualKeyRepositories();

VirtualKeyRepositories f=**new** VirtualKeyRepositories();

System.***out***.println("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.***out***.println("\tWelcome to Virtual Key Repositories ");

System.***out***.println("\t By, Locker Pvt Ltd. \n");

System.***out***.println("\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

**while**(**true**){

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. List Current Files");

System.***out***.println("2. Business Operations");

System.***out***.println("3. Close Application");

System.***out***.println("Please choose one of the following options :");

**int** option1=sc.nextInt();

**switch**(option1){

**case** 1:dp.displayfile();

**break**;

**case** 2:**while**(**true**) {

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. Add a File");

System.***out***.println("2. Delete a File");

System.***out***.println("3. Search for a File");

**int** option2=sc.nextInt();

**switch**(option2){

**case** 1:p.addfile();**break**;

**case** 2:d.deletefile();**break**;

**case** 3:f.searchfile();**break**;

**default** : System.***out***.println("\n Opps! Invalid Input,Re-do the process\n");

**break**;

}

}

**case** 3: System.***out***.println("\n It was nice having you here! See you again. Good bye...");

System.*exit*(0);

**break**;

**default**:

System.***out***.println("\n\n Opps! Invalid Input, Select within the range of 1-3\n");

**break**;

}

}

}}

**Step 2.1:** Writing method to display Initial Menu

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. List Current Files");

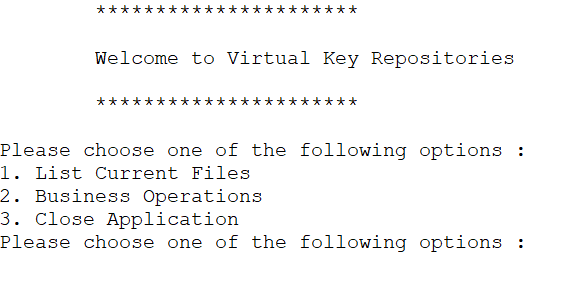
System.***out***.println("2. Business Operations");

System.***out***.println("3. Close Application");

System.***out***.println("Please choose one of the following options :");

**int** option1=sc.nextInt();

**Output:**



**Step 2.2:** Writing method to display Secondary Menu for File Operations

**switch**(option1)

{

**case** 1:dp.displayfile();

**break**;

**case** 2:**while**(**true**) {

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. Add a File");

System.***out***.println("2. Delete a File");

System.***out***.println("3. Search for a File");

**int** option2=sc.nextInt();

**switch**(option2){

**case** 1:p.addfile();**break**;

**case** 2:d.deletefile();**break**;

**case** 3:f.searchfile();**break**;

**default** : System.***out***.println("\n Opps! Invalid Input,Re-do the process\n");

**break**;

}

}

**case** 3: System.***out***.println("\n It was nice having you here! See you again. Good bye...");

System.*exit*(0);

**break**;

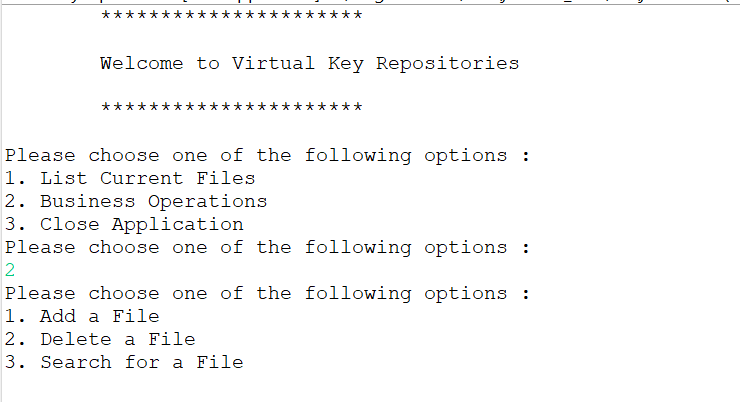
**default**:

System.***out***.println("\n\n Opps! Invalid Input, Select within the range of 1-3\n");

**break**;

}}

**Output:**

****

## **Step 3:** Writing a program in Java to handle Menu options selected by user

**Business** consists methods for -:

* 1. In Business the List of Files is displayed in Ascending order.
  2. In Business to add a file is created.
  3. In Business to delete a file is created.
  4. In Business to search a file is also created.

**Step 3.1:** Writing a method to sort the files in ascending order

**void** displayfile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

File f=**new** File(path);

//display operation

File filenames[]=f.listFiles();

**for**(File ff:filenames)

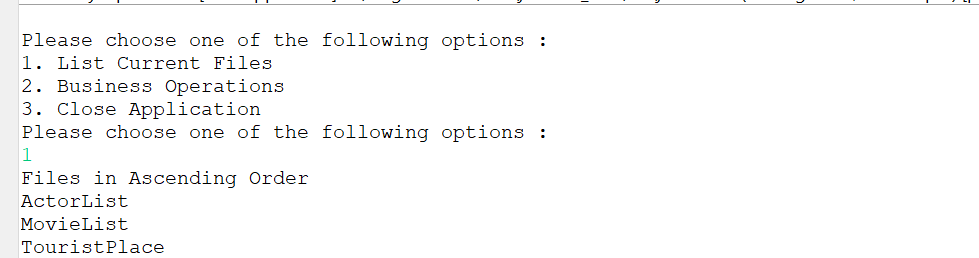
{

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

}

}

**Output:**

****

**Step 3.2:** Writing method to create a file.

**void** addfile() **throws** IOException{

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

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

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

//create a new file

**boolean** b=f.createNewFile();

**if**(b!=**true**)

{

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

}

**else**

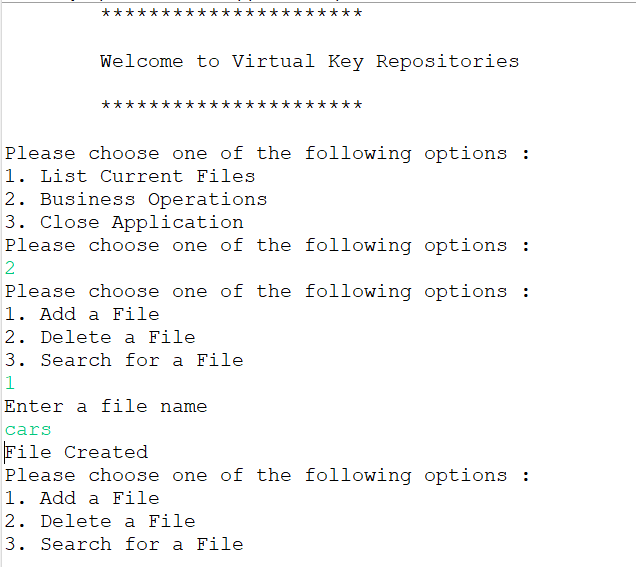
{

System.***out***.println("File Created");

}

}

**Output:**

****

****

**Step 3.3:** Writing method to delete the files

**void** deletefile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter a File Name");

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

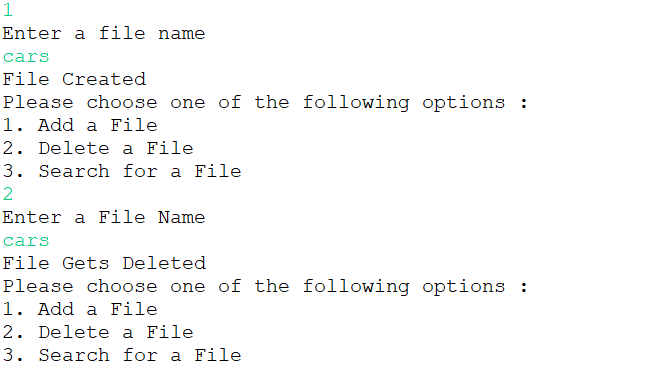
//delete operation

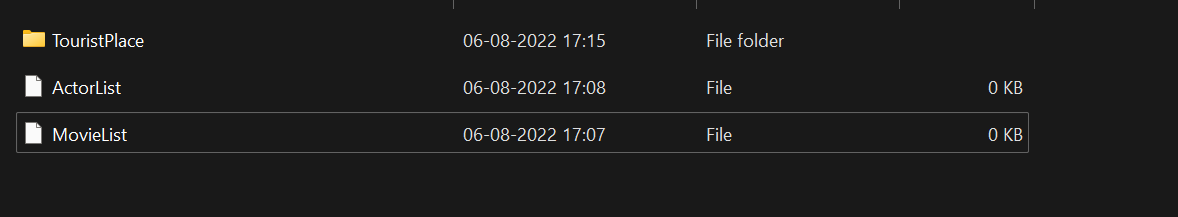
f.delete();

System.***out***.println("File Gets Deleted");

}

**Output:**

****

****

**Step 3.4:** Writing method to search the files

**void** searchfile() {

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter the Filename to Search");

String filename=sc.next();

File f=**new** File(path);

//search operation

**int** flag=0;

File filenames[]=f.listFiles();

**for**(File ff:filenames)

{

**if**(ff.getName().equals(filename))

{

flag=1;

**break**;

}

**else**

{

flag=0;

}

}

**if** (flag==1)

{

System.***out***.println("File is found");

}

**else**

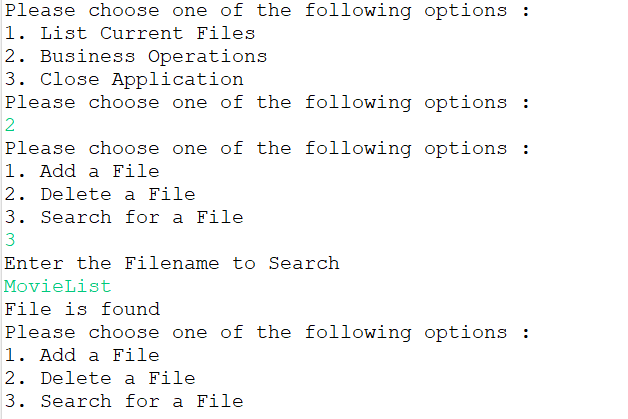
{

System.***out***.println("File is not found");

}

}

**OUTPUT:**



**Step 4:** 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 <commit message>**

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

**git push -u origin master**

**Overall Source Code**

**package** Demo;

**import** java.util.Scanner;

**import** java.io.File;

**import** java.io.IOException;

**public** **class** VirtualKeyRepositories {

**void** displayfile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

File f=**new** File(path);

//display operation

File filenames[]=f.listFiles();

**for**(File ff:filenames)

{

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

}

}

**void** addfile() **throws** IOException{

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

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

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

//create a new file

**boolean** b=f.createNewFile();

**if**(b!=**true**)

{

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

}

**else**

{

System.***out***.println("File Created");

}

}

**void** deletefile(){

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter a File Name");

String filename=sc.next();

String finalpath=path+filename;

File f=**new** File(finalpath);

//delete operation

f.delete();

System.***out***.println("File Gets Deleted");

}

**void** searchfile() {

String path ="D:\\PhaseOneProject\\VirtualKeyRepositories\\";

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

System.***out***.println("Enter the Filename to Search");

String filename=sc.next();

File f=**new** File(path);

//search operation

**int** flag=0;

File filenames[]=f.listFiles();

**for**(File ff:filenames){

**if**(ff.getName().equals(filename)){

flag=1;

**break**;

}

**else**{

flag=0;

}

}

**if** (flag==1){

System.***out***.println("File is found");

}

**else**{

System.***out***.println("File is not found");

}

}

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

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

VirtualKeyRepositories dp=**new** VirtualKeyRepositories();

VirtualKeyRepositories p=**new** VirtualKeyRepositories();

VirtualKeyRepositories d=**new** VirtualKeyRepositories();

VirtualKeyRepositories f=**new** VirtualKeyRepositories();

System.***out***.println("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

System.***out***.println("\tWelcome to Virtual Key Repositories ");

System.***out***.println("\t By, Locker Pvt Ltd. \n");

System.***out***.println("\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

**while**(**true**){

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. List Current Files");

System.***out***.println("2. Business Operations");

System.***out***.println("3. Close Application");

System.***out***.println("Please choose one of the following options :");

**int** option1=sc.nextInt();

**switch**(option1)

{

**case** 1:System.***out***.println("Files in Ascending Order");

dp.displayfile();

**break**;

**case** 2:**while**(**true**) {

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. Add a File");

System.***out***.println("2. Delete a File");

System.***out***.println("3. Search for a File");

**int** option2=sc.nextInt();

**switch**(option2){

**case** 1:p.addfile();**break**;

**case** 2:d.deletefile();**break**;

**case** 3:f.searchfile();**break**;

**default** : System.***out***.println("\n Opps! Invalid Input,Re-do the process\n");

**break**;

}

}

**case** 3: System.***out***.println("\n It was nice having you here! See you again. Good bye...");

System.*exit*(0);

**break**;

**default**:

System.***out***.println("\n\n Opps! Invalid Input, Select within the range of 1-3\n");

**break**;

}

}

}

}

## **Unique Selling Points of the Application**

1. 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.
2. 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.
3. User is also provided the option to write content if they want into the newly created file.
4. 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.
5. The application also allows user to delete folders which are not empty.
6. 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.
7. When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.
   1. Ascending order of folders first which have files sorted in them,
   2. Ascending order of all files and folders inside the “main” folder.
8. 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/folders by different criteria like Last Modified, Type, etc.
* Allowing user to append data to the file.