**LockedMe.Com**

**Virtual Key for Repositories**

**Developed by : Dheeraj Thoke**

**Cohort : MS FSD DEC 2021 Cohort 1**

**COURSE 2**

**Implement OOPS using JAVA with Data Structures and Beyond**

Project GitHub Link:

[**https://github.com/DheerajThoke/JavaFSPhase1Project.git**](https://github.com/DheerajThoke/JavaFSPhase1Project.git)

This document contains sections for:

* [Sprint planning and Task completion](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Sprint_plan)
* [Core concepts used in project](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Core_concepts)
* Source code of Application
* [Flow of the Application](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Flow)
* [Demonstrating the product capabilities, appearance, and user interactions.](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Product_capability)
* [Unique Selling Points of the Application](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#USP)
* [Conclusions](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Conclusions)

## Sprints planning and Task completion

The project is planned to be completed in 2 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.

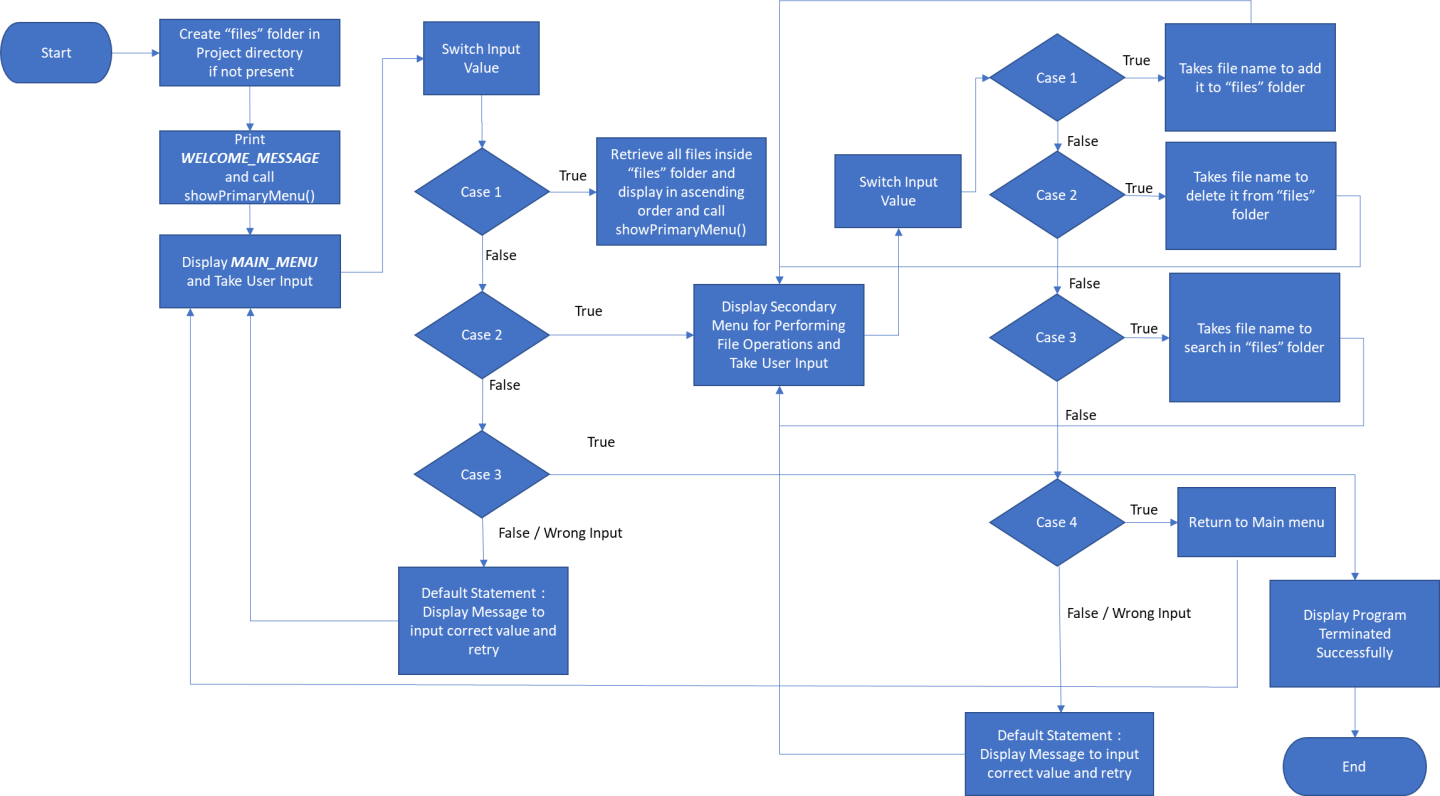
## Core concepts used in project

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

## Source code of Application

|  |
| --- |
| package com.lockedme;  import java.io.File;  import java.io.IOException;  import java.util.Arrays;  import java.util.Scanner;  public class LockedMeMain {  static String directoryPath;  File files;  public LockedMeMain() {  directoryPath = System.getProperty("user.dir");  files = new File(directoryPath+"/files");  if (!files.exists())  files.mkdirs();  System.out.println("Directory Path:- "+ files.getAbsolutePath());  }  private static final String WELCOME\_MESSAGE =  "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+  "\n\*\* Welcome to Lockedme.com (Virtual Key for Your Repositories)"+  "\n\*\* Developed by : Dheeraj Thoke\n"+  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+  "\n\nYou can use this application to :-\n"+  "• Retrieve all file names from the files folder\n"+  "• Search, add, or delete file in the files folder\n"+  "\n\*\*Please enter the complete filename for searching or deleting files\*\*\n"+  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";  private static final String MAIN\_MENU =  "\nMAIN MENU - Enter your choice:-\n"+  "-------------------------------\n"+  "1 -> List all files\n"+  "2 -> Add, Delete or Search\n"+  "3 -> Exit Program";  private static final String SECONDARY\_MENU =  "\nEnter your choice:-\n"+  "-------------------\n"+  "1 -> Add a file\n"+  "2 -> Delete a file\n"+  "3 -> Search a file\n"+  "4 -> Go Back to main menu";  void showPrimaryMenu() {  System.out.println(MAIN\_MENU);  Scanner input = new Scanner(System.in);  try{  int choice = input.nextInt();  switch (choice){  case 1: {  showFiles();  showPrimaryMenu();  break;  }  case 2: {  showSecondaryMenu();  break;  }  case 3: {  System.out.println("Program terminated successfully");  input.close();  System.exit(0);  break;  }  default:{  System.out.println("Please enter a valid option");  showPrimaryMenu();  }  }  }  catch (Exception e){  System.out.println(e.getClass().getName());  System.out.println("Error occured. Please enter a valid option");  showPrimaryMenu();  }  }  void showSecondaryMenu() {  System.out.println(SECONDARY\_MENU);  try{  Scanner in = new Scanner(System.in);  int choice = in.nextInt();  switch (choice){  case 1: {  System.out.println("Enter a file name to add : ");  String filename = in.next().trim();  addFile(filename);  break;  }  case 2: {  System.out.println("Enter a file name to delete : ");  String filename = in.next().trim();  deleteFile(filename);  break;  }  case 3: {  System.out.println("Enter a file name to search : ");  String filename = in.next().trim();  searchFile(filename);  break;  }  case 4: {  System.out.println("Going Back to Main menu");  showPrimaryMenu();  break;  }  default : {  System.out.println("Please enter a valid option");  showSecondaryMenu();  }  }  in.close();  }  catch (Exception e){  System.out.println(e.getClass().getName());  System.out.println("Error occured. Please enter a valid option");  showSecondaryMenu();  }  }  void showFiles() {  if (files.list().length==0)  System.out.println("The folder is empty");  else {  String[] list = files.list();  System.out.println("List of all the files in "+ files +" :\n");  Arrays.sort(list);  for (String str:list) {  System.out.println(str);  }  }  }  void addFile(String File\_name) throws IOException {  File filepath = new File(files +"/"+File\_name);  String[] list = files.list();  for (String file: list) {  if (File\_name.equalsIgnoreCase(file)) {  System.out.println("File " +File\_name + " already exists at " + files);  showSecondaryMenu();  }  }  filepath.createNewFile();  System.out.println("File "+File\_name+" added to "+ files);  showSecondaryMenu();  }  void deleteFile(String File\_name) {  File filepath = new File(files +"/"+File\_name);  String[] list = files.list();  for (String file: list) {  if (File\_name.equals(file)) {  filepath.delete();  System.out.println("File " + File\_name + " deleted from " + files);  showSecondaryMenu();  }  }  System.out.println("Delete Operation failed. File not found.");  showSecondaryMenu();    }  void searchFile(String File\_name) {  String[] list = files.list();  for (String file: list) {  if (File\_name.equals(file)) {  System.out.println("File Found!!! : File " + File\_name + " exists at " + files);  showSecondaryMenu();    }  }  System.out.println("File not found");  showSecondaryMenu();  }  public static void main(String[] args) {    System.out.println(WELCOME\_MESSAGE);  LockedMeMain obj = new LockedMeMain();  obj.showPrimaryMenu();  }  } |

## 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](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Step_1)
2. [Writing a program in Java for the entry point of the application along with the Welcome Message(**LockedMeMain.java**)](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Step_2)
3. [Creating various Strings for displaying primary and secondary menu items](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Step_3)
4. [Writing primary menu method for user to choose primary operation**(showPrimaryMenu())**](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Step_4)
5. [Writing secondary menu method for user to choose from various business level operations **(showSecondaryMenu())**](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#Step_4)
6. Writing all other methods used
7. [Pushing the code to GitHub repository](file:///C:\Users\dheer\Downloads\LockedMe%20-%20Virtual%20Key%20for%20Repositories.docx#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 **LockedMeMain** 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**)

**package** com.lockedme;

**public** **class** LockedMeMain {

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

System.***out***.println(***WELCOME\_MESSAGE***);

LockedMeMain obj = **new** LockedMeMain();

obj.showPrimaryMenu();

}

}

## **Step 3:** Creating various Strings for displaying welcome message, primary and secondary menu items

3.1 ***WELCOME\_MESSAGE :-***

**private** **static** **final** String ***WELCOME\_MESSAGE*** =

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

"\n\*\* Welcome to Lockedme.com (Virtual Key for Your Repositories)"+

"\n\*\* Developed by : Dheeraj Thoke\n"+

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

"\n\nYou can use this application to :-\n"+

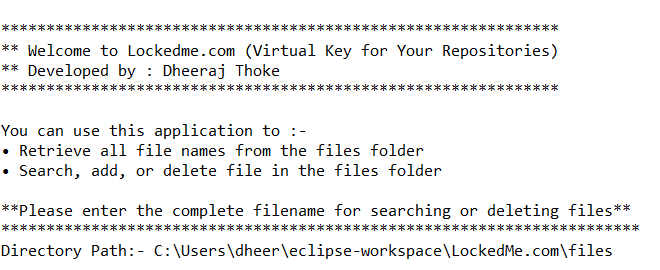
"• Retrieve all file names from the files folder\n"+

"• Search, add, or delete file in the files folder\n"+

"\n\*\*Please enter the complete filename for searching or deleting files\*\*\n"+

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

**Output:**

****

3.2 ***MAIN\_MENU :-***

**private** **static** **final** String ***MAIN\_MENU*** =

"\nMAIN MENU - Enter your choice:-\n"+

"-------------------------------\n"+

"1 -> List all files\n"+

"2 -> Add, Delete or Search\n"+

"3 -> Exit Program";

**Output:**



3.3 ***SECONDARY\_MENU :-***

**private** **static** **final** String ***SECONDARY\_MENU*** =

"\nEnter your choice:-\n"+

"-------------------\n"+

"1 -> Add a file\n"+

"2 -> Delete a file\n"+

"3 -> Search a file\n"+

"4 -> Go Back to main menu";

**Output:**



## **Step 4:** Writing primary menu method for user to choose primary operation (**showPrimaryMenu()**)

**void** showPrimaryMenu() {

System.***out***.println(***MAIN\_MENU***);

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

**try**{

**int** choice = input.nextInt();

**switch** (choice){

**case** 1: {

showFiles();

showPrimaryMenu();

**break**;

}

**case** 2: {

showSecondaryMenu();

**break**;

}

**case** 3: {

System.***out***.println("Program terminated successfully");

input.close();

System.*exit*(0);

**break**;

}

**default**:{

System.***out***.println("Please enter a valid option");

showPrimaryMenu();

}

}

}

**catch** (Exception e){

System.***out***.println(e.getClass().getName());

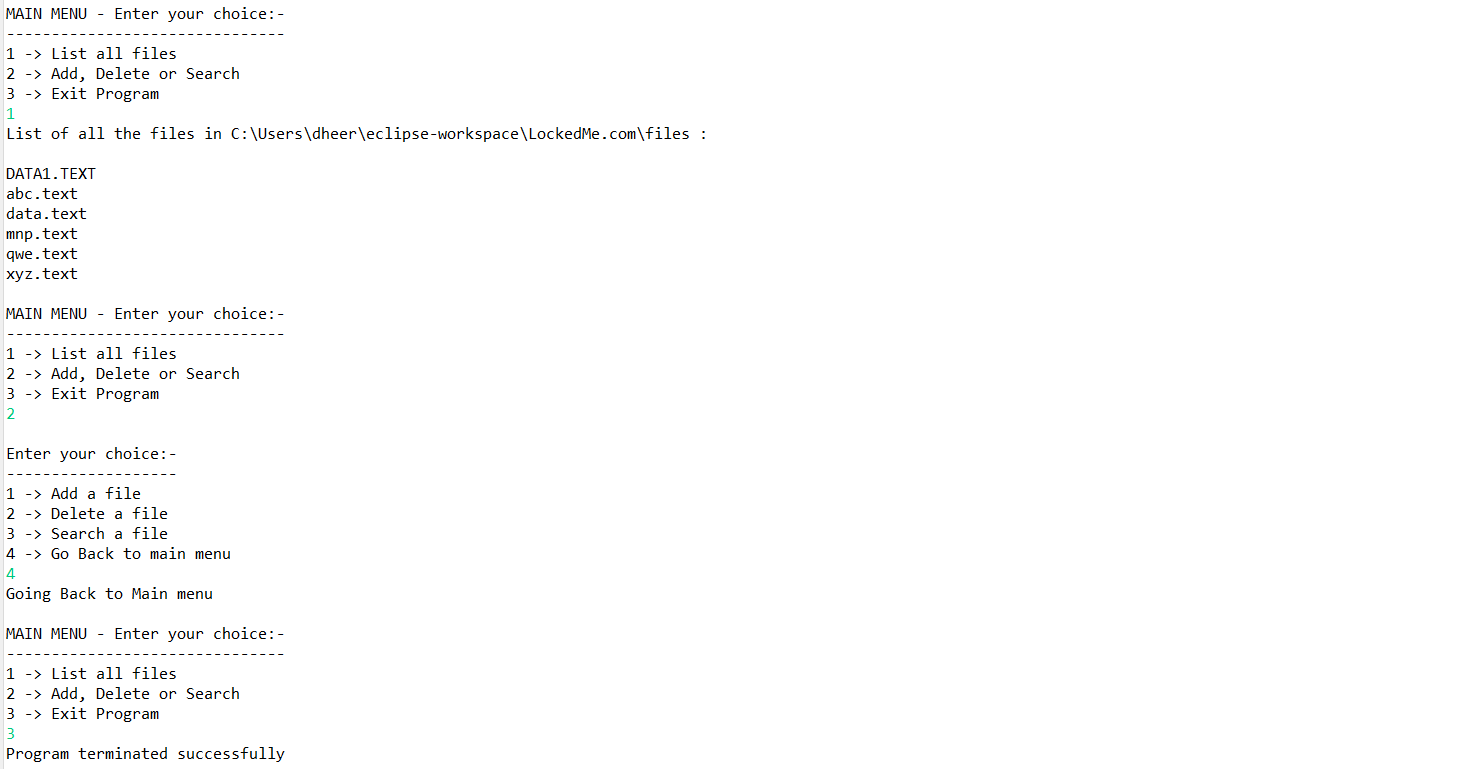
System.***out***.println("Error occured. Please enter a valid option");

showPrimaryMenu();

}

}

**Output:**

****

## **Step 5:** Writing secondary menu method for user to choose from various business level operations(**showSecondaryMenu()**)

**void** showSecondaryMenu() {

System.***out***.println(***SECONDARY\_MENU***);

**try**{

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

**int** choice = in.nextInt();

**switch** (choice){

**case** 1: {

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

String filename = in.next().trim();

addFile(filename);

**break**;

}

**case** 2: {

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

String filename = in.next().trim();

deleteFile(filename);

**break**;

}

**case** 3: {

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

String filename = in.next().trim();

searchFile(filename);

**break**;

}

**case** 4: {

System.***out***.println("Going Back to Main menu");

showPrimaryMenu();

**break**;

}

**default** : {

System.***out***.println("Please enter a valid option");

showSecondaryMenu();

}

}

in.close();

}

**catch** (Exception e){

System.***out***.println(e.getClass().getName());

System.***out***.println("Error occured. Please enter a valid option");

showSecondaryMenu();

}

}

****

## **Step 6:** Writing all other methods used

**Step 6.1:** Writing method to add file(**addFile(String File\_name)**)

**void** addFile(String File\_name) **throws** IOException {

File filepath = **new** File(files +"/"+File\_name);

String[] list = files.list();

**for** (String file: list) {

**if** (File\_name.equalsIgnoreCase(file)) {

System.***out***.println("File " +File\_name + " already exists at " + files);

showSecondaryMenu();

}

}

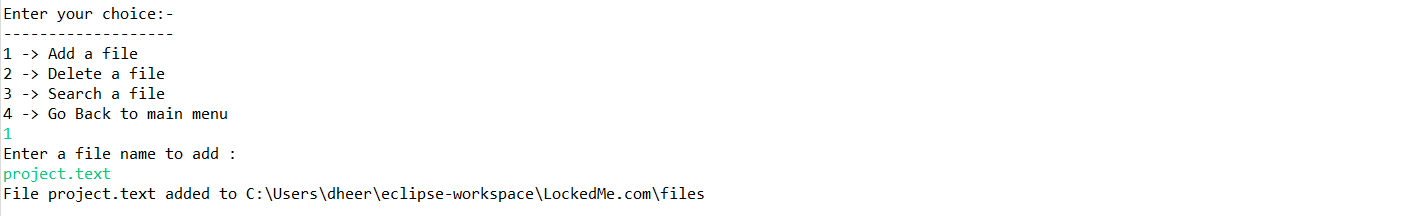
filepath.createNewFile();

System.***out***.println("File "+File\_name+" added to "+ files);

showSecondaryMenu();

}

**Output:**

****

**Step 6.2:** Writing method to delete file(**deleteFile(String File\_name)**)

**void** deleteFile(String File\_name) {

File filepath = **new** File(files +"/"+File\_name);

String[] list = files.list();

**for** (String file: list) {

**if** (File\_name.equals(file)) {

filepath.delete();

System.***out***.println("File " + File\_name + " deleted from " + files);

showSecondaryMenu();

}

}

System.***out***.println("Delete Operation failed. File not found.");

showSecondaryMenu();

}

**Output:**

## 

**Step 6.3:** Writing method to search file(**searchFile(String File\_name)**)

**void** searchFile(String File\_name) {

String[] list = files.list();

**for** (String file: list) {

**if** (File\_name.equals(file)) {

System.***out***.println("File Found!!! : File " + File\_name + " exists at " + files);

showSecondaryMenu();

}

}

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

showSecondaryMenu();

}

**Output:**

****

**Step 6.4:** Writing method to show files(**showFiles()**)

**void** showFiles() {

**if** (files.list().length==0)

System.***out***.println("The folder is empty");

**else** {

String[] list = files.list();

System.***out***.println("List of all the files in "+ files +" :\n");

Arrays.*sort*(list);

**for** (String str:list) {

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

}

}

}

**Output:**

## 

**Step 6.5:** Creating class constructor for currect directory path and to make new directory if not available(**LockedMeMain()**)

**public** LockedMeMain() {

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

files = **new** File(*directoryPath*+"/files");

**if** (!files.exists())

files.mkdirs();

System.***out***.println("Directory Path:- "+ files.getAbsolutePath());

}

**Output:**

****

## **Step 7:** 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 remote repository using the following command:

**git remote add origin** [**https://github.com/DheerajThoke/JavaFSPhase1Project.git**](https://github.com/DheerajThoke/JavaFSPhase1Project.git)

**git push -u origin main**

**git branch -M main**

## 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 also displays message if folder is empty.
3. 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.
4. 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.