:: Project ::

Virtual Key for Repositories
(LockedMe)

Submitted By:

Rajan Kumar

Contents

This document contains sections for**:**

Note : The code for this project is hosted at **:** https://github.com/RajanKr004/lockedMe

1. Sprint Planning and Task Completion

The project is planned to be completed in 1 sprint.

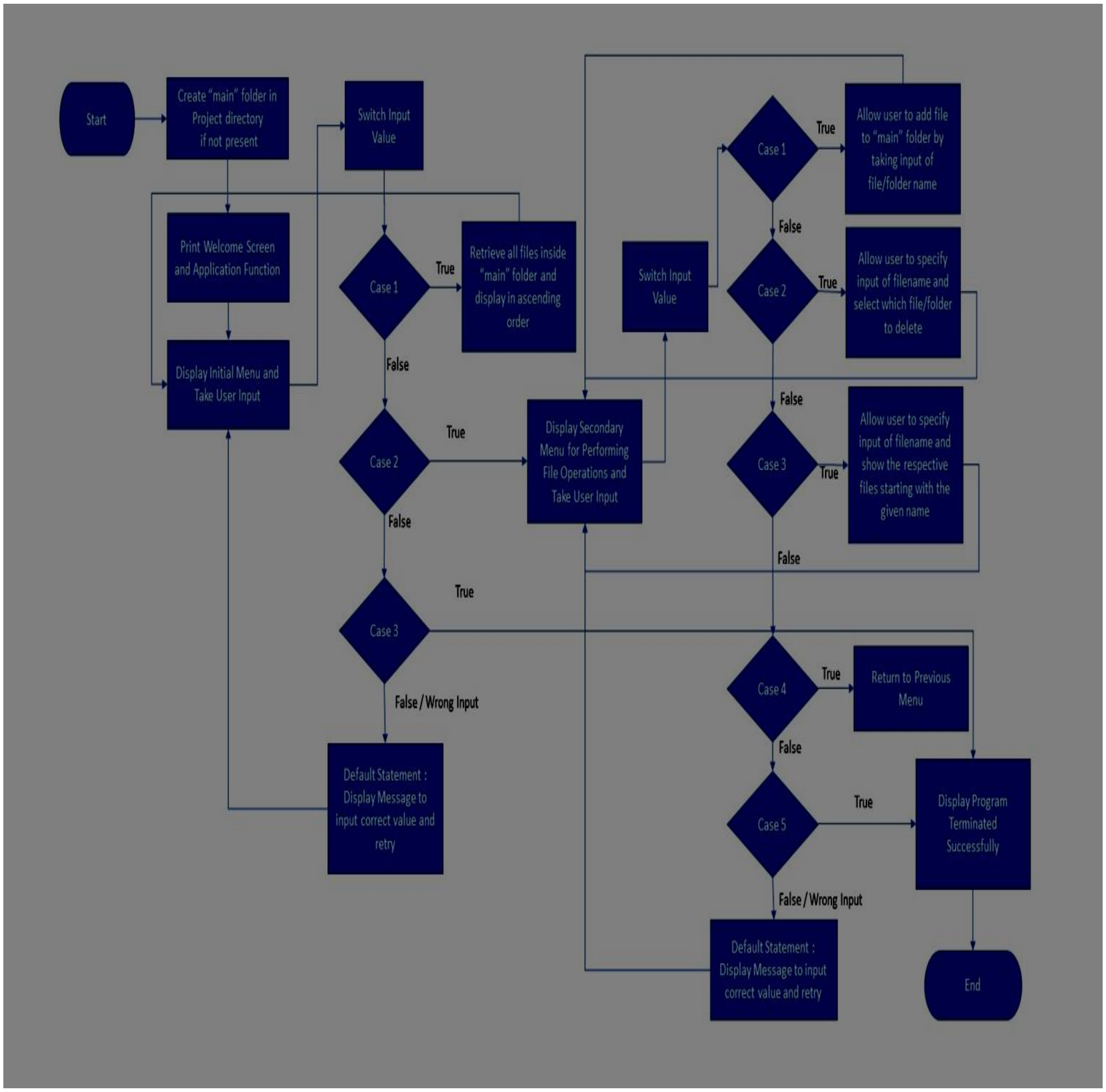
Tasks assumed to be completed in the sprints are :

- 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.

2. Core concepts used in project :

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

3. Flow of the Application


Start
Create "main" folder in Project directory if not present
Print Welcome Screen and Application Function
Display Initial Menu and Take User Input
Switch Input Value
Case 1
True
Retrieve all files inside "main" folder and display in ascending order
False
Case 2
True
Display Secondary 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 "main" 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 Previous Menu
False
Case 5
True
Display Program Terminated Successfully
False / Wrong Input
Default Statement : Display Message to input correct value and retry
End

4. Demonstrating the product capabilities, appearance, and user interactions

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

1 Creating the project in Eclipse

2 Writing program in Java for the entry point of the application (**LockedMe.java**)

3 Writing program in Java to display Menu options available for the user

4 Writing program in Java to Retrieve All files inside the Main Folder

5 Writing program in Java to perform the File operations as specified by user

5.1 Showing options for the file operations
5.2 Adding Files and Folders
5.3 Deleting Files and Folders
5.4 searching any Files and Folder
5.5 Thanking the user for using the Application

6 Pushing the code to GitHub repository

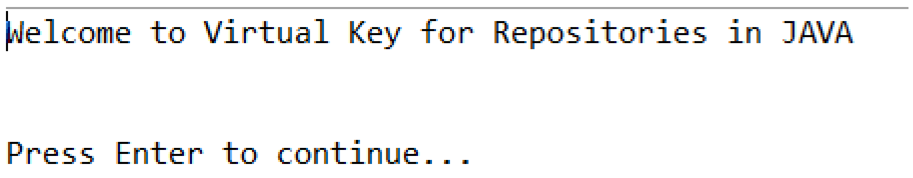
Step 1: Creating a new project in Eclipse

- 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 (**LockedMe.java**)

```
public class lockedMe {

	public static void main(String[] args)throws IOException, InterruptedException
	{
		Scanner sc = new Scanner(System.in);
		int choice,Fchoice,OPchoice,  fc=0,j, count=0;
		String st;
		File mFolder =new File("C:\\Users\\Rajan Kumar\\Desktop\\JAVAMAIN");
		mFolder.mkdir();
		String location = "C:\\Users\\Rajan Kumar\\Desktop\\JAVAMAIN";

		System.out.println("Welcome to Virtual Key for Repositories in 
JAVA\n\n");
		System.out.println("Press Enter to continue...");
	    System.in.read();


	    System.out.print("\u000C");
```

Output :



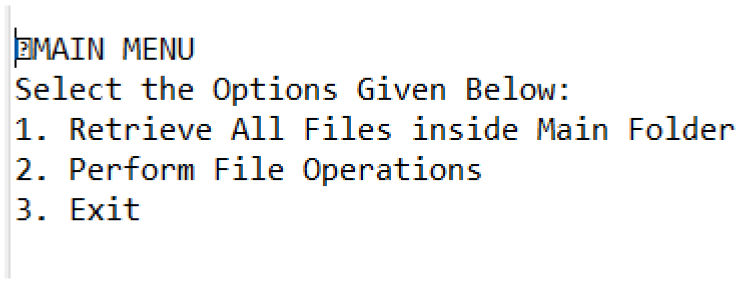
```
Welcome to Virtual Key for Repositories in JAVA


Press Enter to continue...
```

Step 3: Writing a program in Java to display Menu options available for the user

```
do {
		choice=0;
	    System.out.println("MAIN MENU");
	    System.out.println("Select the Options Given Below:");
	    System.out.println("1. Retrieve All Files inside Main Folder");
	    System.out.println("2. Perform File Operations");
```

```
        System.out.println("3. Exit\n");
        choice= sc.nextInt();
```

Output:

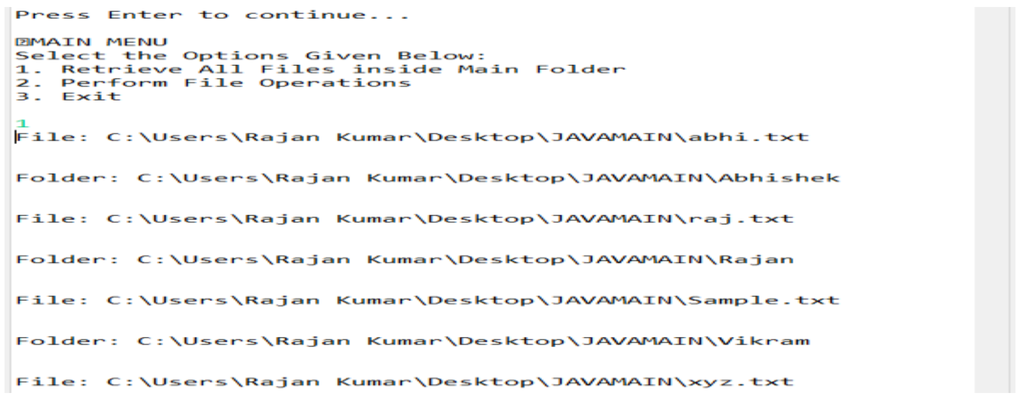


```
⍰MAIN MENU
Select the Options Given Below:
1. Retrieve All Files inside Main Folder
2. Perform File Operations
3. Exit
```

Step 4: Writing a program in Java to Retrieve All files inside the Main Folder

```
case 1:
          File Fobj = new File("C:\\Users\\Rajan Kumar\\Desktop\\JAVAMAIN");
                 File Farray[] = Fobj.listFiles();
          for(int i=0;i<Farray.length;i++) {
                 if(Farray[i].isFile()) {
                        System.out.println("File: "+Farray[i] +"\n\n");
                 }
                 else if(Farray[i].isDirectory()) {
                        System.out.println("Folder: "+Farray[i]+"\n\n");
                 }
          }
          break;
```

Output:



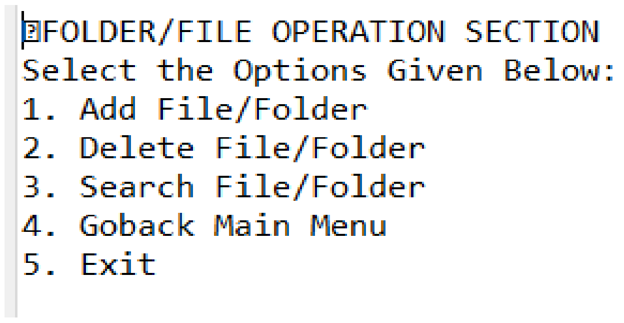
```
Press Enter to continue...

⍰MAIN MENU
Select the Options Given Below:
1. Retrieve All Files inside Main Folder
2. Perform File Operations
3. Exit

1
File: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\abhi.txt

Folder: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\Abhishek

File: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\raj.txt

Folder: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\Rajan

File: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\Sample.txt

Folder: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\Vikram

File: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\xyz.txt
```

Step 5: Writing a program in Java to perform the File operations as specified by user

Step 5.1: Writing Program for Showing options for the file operations

```
case 2:
        do {
                count=0;
                System.out.print("\u000C");
        System.out.println("FOLDER/FILE OPERATION SECTION");
        System.out.println("Select the Options Given Below:");
            System.out.println("1. Add File/Folder");
            System.out.println("2. Delete File/Folder");
            System.out.println("3. Search File/Folder");
            System.out.println("4. Goback Main Menu");
            System.out.println("5. Exit\n");
            Fchoice= sc.nextInt();
```

Output:



```
□FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit
```

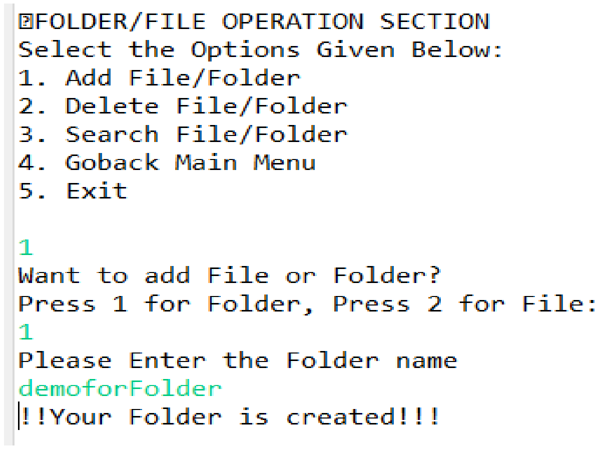
Step 5.2: Writing Program for Adding Files and Folders

```
switch(Fchoice) {
            case 1:

               System.out.println("Want to add File or Folder?");
               System.out.println("Press 1 for Folder, Press 2 for File:");
               OPchoice= sc.nextInt();
               switch(OPchoice) {
               case 1:
                       System.out.println("Please Enter the Folder name");
                       String str1 = sc.next();
```

```
                        File addfolder = new File(location+"\\"+str1);
                        if(addfolder.exists()) {
                                System.out.println("Already exits !! Please Enter 
Again\n");
                        }
                        else {
                                addfolder.mkdir();
                                System.out.println("!!Your Folder is created!!!\n");
                        }
                        break;
                case 2:
                        System.out.println("Please Enter the File name with exten-
sion");
                        String str2 = sc.next();
                        File addfile = new File(location+"\\"+str2);
                        if(addfile.exists()) {
                                System.out.println("Already exits !! Please Enter 
Again\n");

                                }else {
                                        try {
                                                addfile.createNewFile();
                                                System.out.println("!!Your file is cre-
ated!!!\n");

                                        }catch(IOException e) {
                                                e.printStackTrace();
                                        }
                                }

                break;
                default:
                        System.out.println("Please Enter Correct Value!!!!!\n");

                }

                break;
```

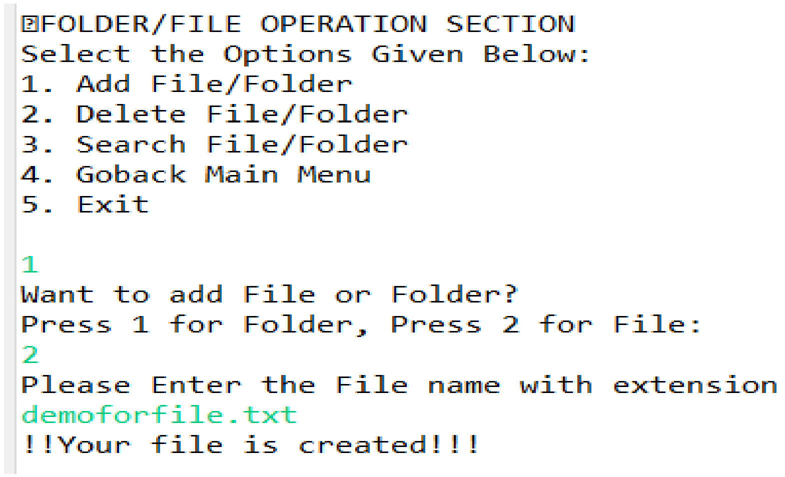
Output:

1. **Adding folder :**



```
▯FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

1
Want to add File or Folder?
Press 1 for Folder, Press 2 for File:
1
Please Enter the Folder name
demoforFolder
!!Your Folder is created!!!
```

2. **Adding File :**



```
▯FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

1
Want to add File or Folder?
Press 1 for Folder, Press 2 for File:
2
Please Enter the File name with extension
demoforfile.txt
!!Your file is created!!!
```

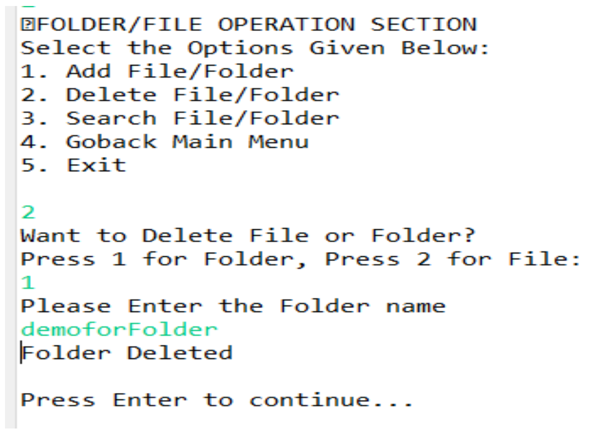
Step 5.3 : Writing Program for Deleting Files and Folders

```
case 2:

                        fc = 0;
                System.out.println("Want to Delete File or Folder?");
                System.out.println("Press 1 for Folder, Press 2 for File:");
                fc = sc.nextInt();
```

```
                    switch(fc) {
                        case 1:
                        System.out.println("Please Enter the Folder name");
                        st = sc.next();
                        File delFolder = new File(location+"\\"+st);
                        if(delFolder.exists()) {
                            delFolder.delete();
                            System.out.println("Folder Deleted\n");
                        }
                        else {
                            System.out.println("!!Folder not Found!!!\n");
                        }
                        break;
                        case 2:
                        System.out.println("Please Enter the File name with exten-
sion");

                        st = sc.next();
                        File delFile = new File(location+"\\"+st);
                        if(delFile.exists()) {
                            delFile.delete();
                            System.out.println("File Deleted\n");
                            }else {
                                System.out.println("!!File not Found!!!\n");
                            }
                    break;
                    default:
                        System.out.println("Please Enter Correct Value!!!!!\n");
                    }
                    break;
```

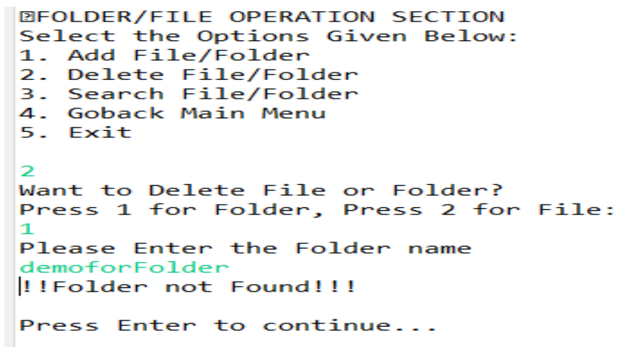
Output:

1. **Deleting folder :**



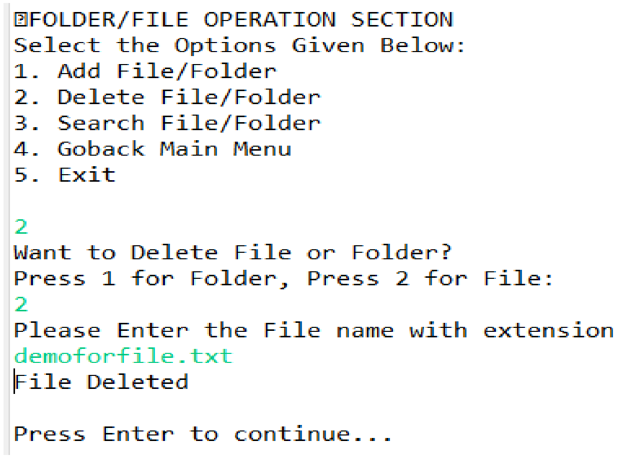
```
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

2
Want to Delete File or Folder?
Press 1 for Folder, Press 2 for File:
1
Please Enter the Folder name
demoforFolder
Folder Deleted

Press Enter to continue...
```

**** When folder is not there !!**



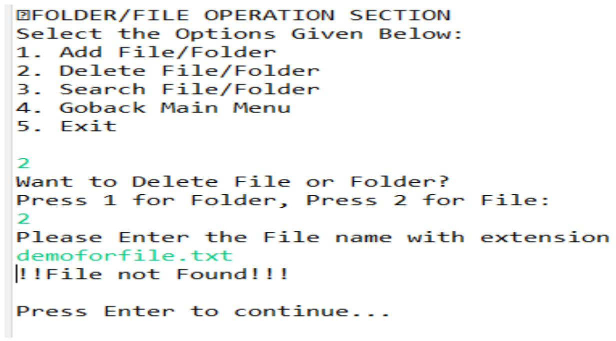
```
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

2
Want to Delete File or Folder?
Press 1 for Folder, Press 2 for File:
1
Please Enter the Folder name
demoforFolder
!!Folder not Found!!!

Press Enter to continue...
```

2. **Deleting File :**



```
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

2
Want to Delete File or Folder?
Press 1 for Folder, Press 2 for File:
2
Please Enter the File name with extension
demoforfile.txt
File Deleted

Press Enter to continue...
```

**** When file is not there !!**

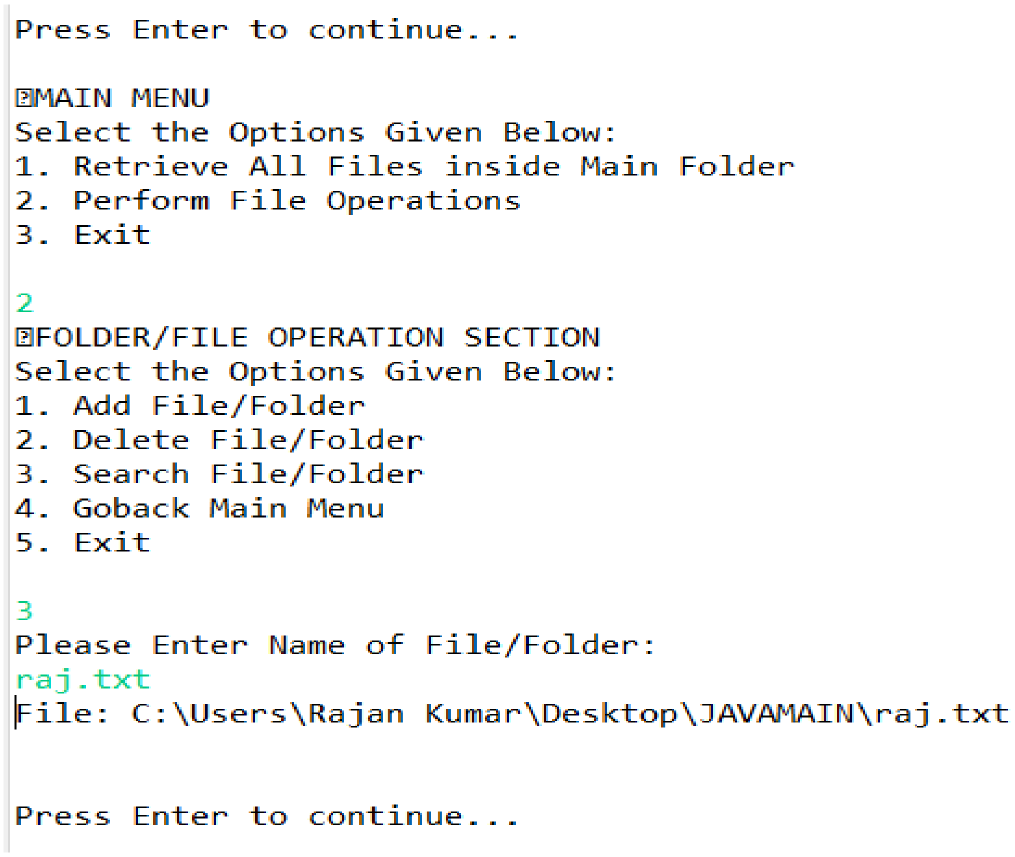


```
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

2
Want to Delete File or Folder?
Press 1 for Folder, Press 2 for File:
2
Please Enter the File name with extension
demoforfile.txt
!!File not Found!!!

Press Enter to continue...
```

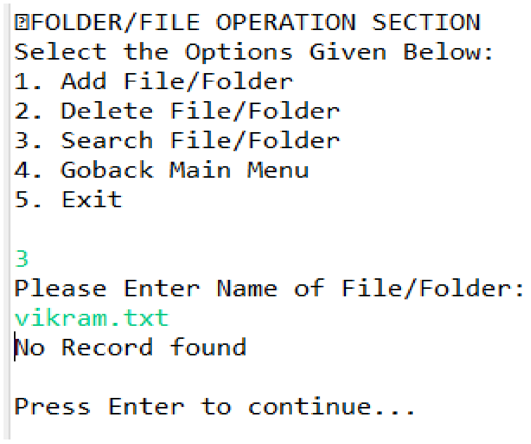
Step 5.4 : Writing Program for searching any Files and Folder

```
        case 3:
                    System.out.println("Please Enter Name of File/Folder:");
                    st = sc.next();
                    File fsearch = new File("C:\\Users\\Rajan Kumar\\Desktop\\JA-
VAMAIN");
                        File FSarray[] = fsearch.listFiles();
                    for(int i=0;i<FSarray.length;i++) {
                        if(FSarray[i].getName().startsWith(st)) {
                            count++;
                            if(FSarray[i].isFile()) {
                                System.out.println("File: "+FSar-
ray[i]+"\n\n");
                            }
                            else if(FSarray[i].isDirectory()) {
                                System.out.println("Folder: "+FSar-
ray[i]+"\n\n");
                            }
                        }
                    }
                    if(count==0)
                        System.out.println("No Record found\n");
                    break;
                 case 4:
                    break;
                 case 5:
                    System.out.println("Thanks for using my application");
                    System.exit(1);
                    break;

                 default:
                     System.out.println("Wrong input");

                 }

                 System.out.println("Press Enter to continue...");
                 System.in.read();
              }while(Fchoice!=4);
              break;
```

Output:


```
Press Enter to continue...

MAIN MENU
Select the Options Given Below:
1. Retrieve All Files inside Main Folder
2. Perform File Operations
3. Exit

2
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

3
Please Enter Name of File/Folder:
raj.txt
File: C:\Users\Rajan Kumar\Desktop\JAVAMAIN\raj.txt


Press Enter to continue...
```

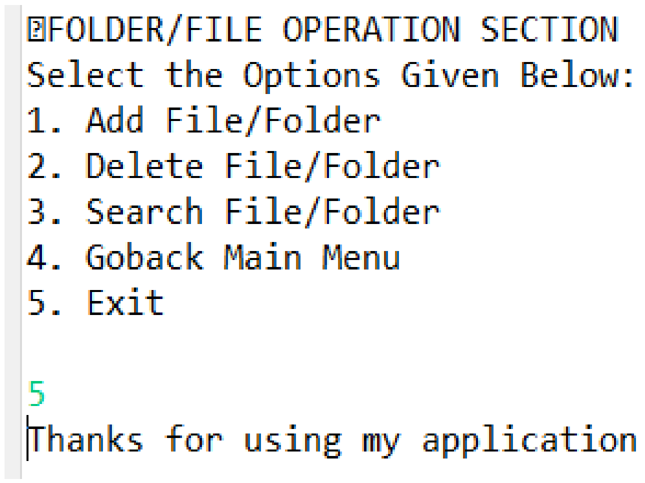
**** When Files and Folders are not present**


```
FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

3
Please Enter Name of File/Folder:
vikram.txt
No Record found

Press Enter to continue...
```

Step 5.5 : Writing Program for Thanking the user

```
case 3:
                System.out.println("Thanks for using my application");
                System.exit(1);
                default:
                         System.out.println("Wrong input");
             }
             }while(choice!=3);
          sc.close();
```

output :



```
?FOLDER/FILE OPERATION SECTION
Select the Options Given Below:
1. Add File/Folder
2. Delete File/Folder
3. Search File/Folder
4. Goback Main Menu
5. Exit

5
Thanks for using my application
```

Step 6: 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

5. 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.

   7.1. Ascending order of folders first which have files sorted in them,
   7.2. 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.

6. 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.