**Virtual Key for Your Repositories**

**Document contains:**

* Project and developer details
* Sprints planned and the tasks achieved in them
* Algorithms and flowcharts of the application
* Core concepts used in the project
* Links to the GitHub repository to verify the project completion
* Unique Selling Points of the Application
* Conclusions

**Project and developer details**

**Project objective:**

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**The flow and features of the application:**

Plan more than two sprints to complete the application

Document the flow of the application and prepare a flow chart

List the core concepts and algorithms being used to complete this application

Code to display the welcome screen. It should display:

Application name and the developer details

The details of the user interface such as options displaying the user interaction information

Features to accept the user input to select one of the options listed

The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it

The second option should return the details of the user interface such as options displaying the following:

Add a file to the existing directory list

You can ignore the case sensitivity of the file names

Delete a user specified file from the existing directory list

You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list

Return a message if FNF (File not found)

Search a user specified file from the main directory

You can add the case sensitivity on the file name to retrieve the correct file

Display the result upon successful operation

Display the result upon unsuccessful operation

Option to navigate back to the main context

There should be a third option to close the application

Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

**Developer Details**: P Praveen Kumar

Ppraveen62.ppk@gmail.com

9502646206

**Sprints planned and the tasks achieved in them**

There are three sprints for this project

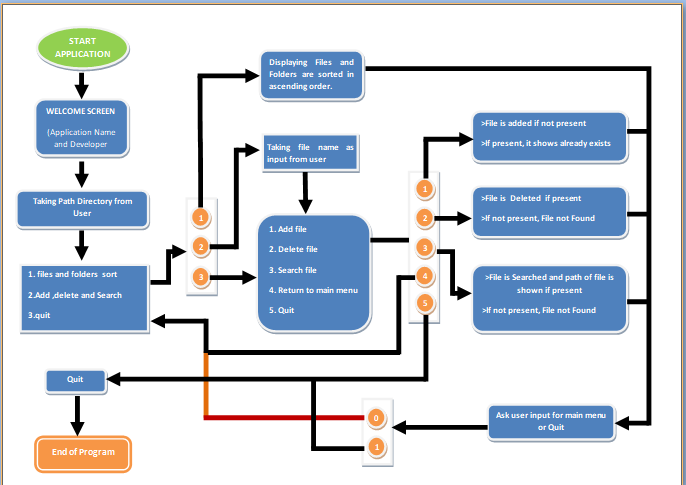
Sprint-1: Studied the Features of application and prepared the flow chart and Git Repository.

Sprint-2: Written java program for welcome screen and All options, solving errors and exceptions.

Sprint-3: Final testing of program using different inputs and pushing it to GitHub.

Algorithms and flowcharts of the application

**Flow chart:**



1. Created a project in eclipse VirtualKeyRepo

2. Created a package virtualKeyRepo

3. First created a class MainMethods.java ,It is abstract classs which contains all methods and static variables which are being used in all methods of project.

**Code1:**

Package virtualKeyRepo;

public abstract class MainMethods {

static String *path*;

static int *option*;

static String *fname*;

static int *i*;

public abstract void intro();

public abstract void intro1();

public abstract void menu();

public abstract void sort();

public abstract void subMenu();

public abstract void subAdd();

public abstract void subDelete();

public abstract void subSearch();

public abstract void backADS();

public abstract void backRun();

public abstract void backSubMenu();

public abstract void backMenu();

}

4. created a new class DisplayMenu which extends MainMethods here all methods are implemented by me.

**Code2:**

package virtualKeyRepo;

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

//display of welcome screen and first menu

public class DisplayMenu extends MainMethods {

public void intro() {

String outlineBorder="--------------------------------------------------------------------------------------------------";

String appName="VIRTUAL KEY FOR YOUR REPOSIITORIES";

String devName="-ppk";

System.out.println();

System.out.println(outlineBorder);

System.out.println();

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

System.out.println(" | "+appName+" |");

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

System.out.println(" | "+devName+" |");

System.out.println();

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

System.out.println();

}

public void intro1() {

Scanner sc =new Scanner(System.in);

System.out.println();

System.out.println(" Info:-");

System.out.println(" Here you can Sort, Add, Delete and Search opreration of fIles in your given path Directory.");

System.out.println();

System.out.print(" Give the Path of Directory:- ");

DisplayMenu.path=sc.next();

System.out.println();

}

public void menu() {

Scanner sc =new Scanner(System.in);

System.out.println();

System.out.println(" >>>>>>>>>>>>>>>>>>>>");

System.out.println();

System.out.println(" File operations are : \n 1. Sort Files in Asscending Order\n 2. Add, Delete, Sreach a FILe\n 3. Quit");

System.out.println();

backMenu();

switch(DisplayMenu.option) {

case 1:

sort();

backADS();

break;

case 2:

subMenu();

break;

case 3:

System.out.print(" \*\*\*Exited Sucessfully\*\*\* ");

i=1;

break;

}

}

public void sort() {

File obj =new File(DisplayMenu.path); //giving a directory

String [] file= obj.list();

//Arrays.sort(file);

//we are creating list array using list() method

System.out.println();

System.out.println(" The files and folders in "+DisplayMenu.path+" are >");

System.out.println();

for (String x:file) { //Accessing list from each.

System.out.println(" "+x);

}

}

public void subMenu() {

Scanner sc =new Scanner(System.in);

System.out.println();

System.out.print(" Enter File Name for which you want do operations in existing Directory,\n (" +DisplayMenu.path+" ) :- ");

DisplayMenu.fname=sc.nextLine();

System.out.println();

System.out.println(" >>>>>>>>>>>>>>>>>>>>");

System.out.println();

System.out.println(" 1. Add file\n 2. Delete file\n 3. Search file\n 4. Return to Main Menu\n 5. Quit\n");

System.out.println();

backSubMenu();

switch(DisplayMenu.option) {

case 1:

subAdd();

backADS();

break;

case 2:

subDelete();

backADS();

break;

case 3:

subSearch();

backADS();

break;

case 4:

menu();

break;

case 5:

System.out.print(" \*\*\*Exited Sucessfully\*\*\* ");

i=1;

break;

}

}

public void subAdd() {

File obj =new File(DisplayMenu.path+"//"+DisplayMenu.fname);

try {

if(obj.createNewFile()) {

System.out.println(" File is Added");

}

else {

System.out.println(" File already exists");

}

} catch (IOException e) {

e.printStackTrace();

}

}

public void subDelete() {

File obj =new File(DisplayMenu.path+"//"+DisplayMenu.fname);

if(obj.delete()) {

System.out.println(" File is deleted");

}

else {

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

}

}

public void subSearch() {

File obj =new File(DisplayMenu.path+"//"+DisplayMenu.fname); //giving a directory

boolean sreach =obj.isFile(); //true since file exists

if(sreach==true) {

System.out.println(" "+obj.getAbsolutePath());

}else {

System.out.println(" File not Found");

}

}

public void backADS() {

int j=0;

while(j==0) {

System.out.println();

System.out.print(" Enter 0 for main menu or 1 to exit? ");

Scanner sc=new Scanner(System.in);

try {

int k=sc.nextInt();

if(k==0) {

menu();

j+=1;

}

if (k==1) {

j+=1;

System.out.print(" \*\*\*Exited Sucessfully\*\*\* ");

DisplayMenu.i=1;

}

if(k!=0 &&k!=1) {

System.out.println(" wrong input");

}

} catch (Exception e) {

System.out.println();

System.out.println(" Please check input ");

System.out.println();

}

}

}

public void backRun() {

int j=0;

while(j==0) {

System.out.println();

System.out.print(" Enter 0 for main menu or 1 to exit? ");

Scanner sc=new Scanner(System.in);

try {

int k=sc.nextInt();

if(k==0) {

intro1();

j+=1;

}

if (k==1) {

j+=1;

System.out.print(" \*\*\*Exited Sucessfully\*\*\* ");

DisplayMenu.i=1;

}

if(k!=0 &&k!=1) {

System.out.println(" wrong input");

}

} catch (Exception e) {

System.out.println();

System.out.println(" Please check input ");

System.out.println();

}

}

}

public void backMenu() {

int j=0;

while(j==0) {

System.out.println();

System.out.print(" Select one of the option (1/2/3):- ");

Scanner sc=new Scanner(System.in);

try {

DisplayMenu.option=sc.nextInt();

if(DisplayMenu.option==1||DisplayMenu.option==2||DisplayMenu.option==3) {

j+=1;

}

else {

System.out.println(" wrong input");

}

} catch (Exception e) {

System.out.println();

System.out.println(" Please check input ");

System.out.println();

}

}

}

public void backSubMenu() {

int j=0;

while(j==0) {

System.out.println();

System.out.print(" Select one of the option (1/2/3/4/5):- ");

Scanner sc=new Scanner(System.in);

try {

DisplayMenu.option=sc.nextInt();

if(DisplayMenu.option==1||DisplayMenu.option==2||DisplayMenu.option==3||DisplayMenu.option==4||DisplayMenu.option==5) {

j+=1;

}

else {

System.out.println(" wrong input");

}

} catch (Exception e) {

System.out.println();

System.out.println(" Please check input ");

System.out.println();

}

}

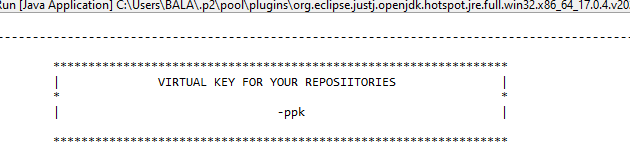
}

}

5. Every method is implemented according to the need in the project let see at output

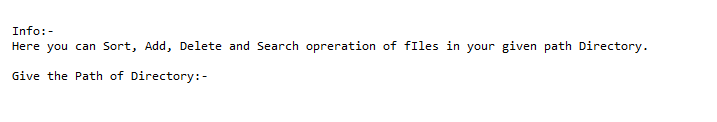
**intro();**

Here it is the welcome screen showing project name and developer name.



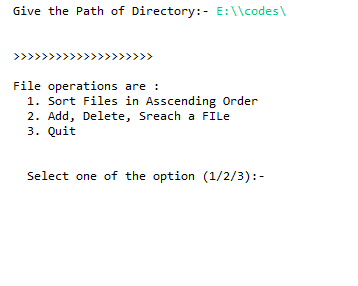
**intro1();**

Here it is taking input from the user for path of directory for operations given., if path exists move further else ask for exit or continue if continue again asks the path.



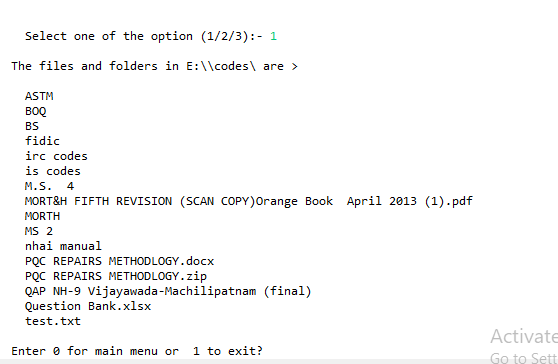
**menu();**

You can give options according to give below and this methods calls most methods.



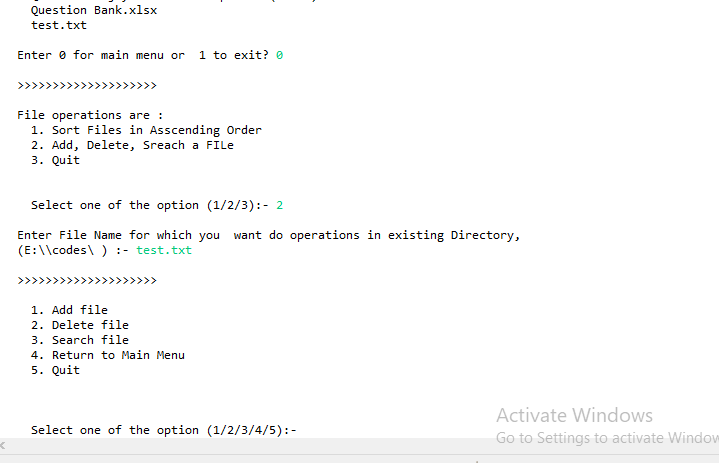
**sort();**

Shows files and folders in ascending order.

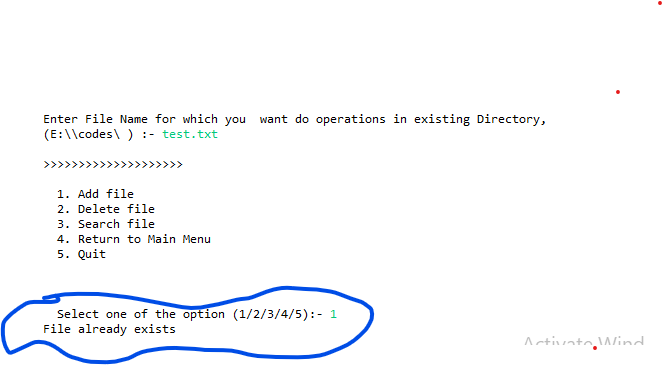


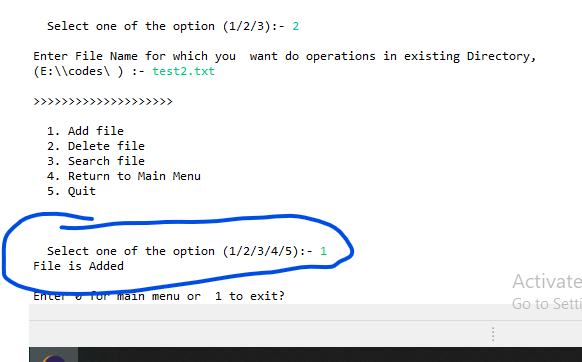
**subMenu();**

If the option is 2 from menu, it goes to another menu as shown below before we should have to give the file name for the further operations.

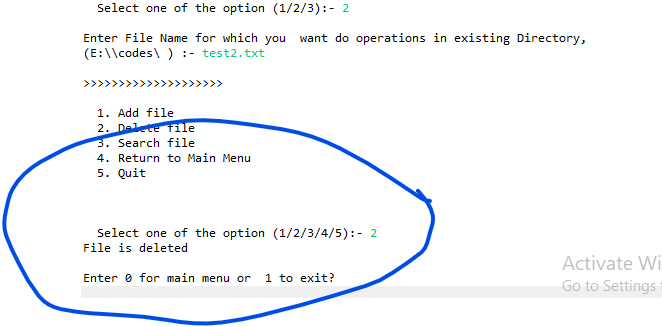


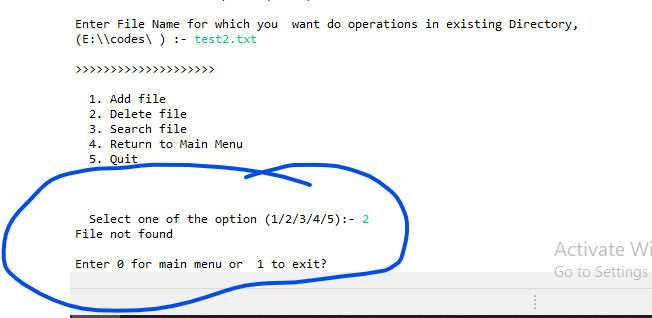
**subAdd();**



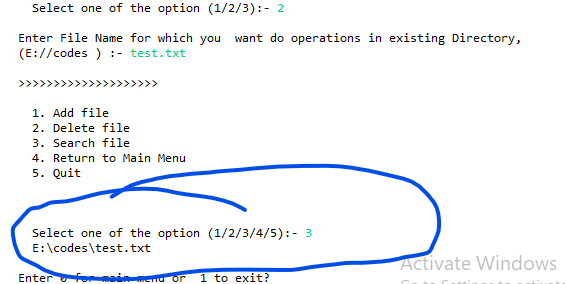


**subDelete();**





**subSearch();**



**back\_\_();**

This methods are being used for not getting stop if any exception in case of wrong input and ask for further check and give right input.

5. Created another class Run.java this class contain main method where intro(), intro1() and menu() method is being called.

**Code3:**

package virtualKeyRepo;

import java.io.File;

import java.util.Scanner;

public class Run {

public static void main(String[] args) {

// TODO Auto-generated method stub

DisplayMenu obj=new DisplayMenu();

obj.intro();

obj.intro1();

DisplayMenu.i=0;

while(DisplayMenu.i==0) {

File ob=new File(DisplayMenu.path); //giving a directory

boolean search =ob.isDirectory();

if (search==true) {

obj.menu();

DisplayMenu.i+=1;

}

else {

System.out.println(" Path not Exists ");

obj.backRun();

}

}

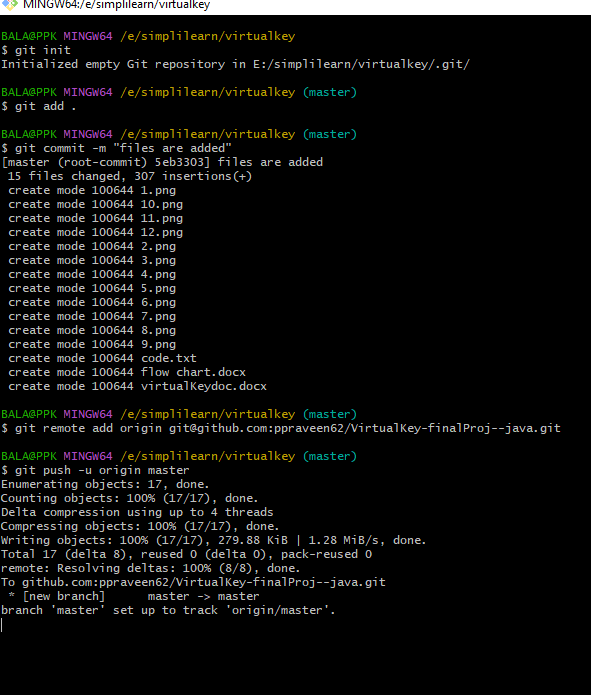
}

}

**Output**

[**Click here**](output.txt)

**Pushing to GitHub**



**Core concepts used in the project**

File handling, Abstract, exceptions, collections, and sorting techniques.

**Links to the GitHub repository to verify the project completion**

<https://github.com/ppraveen62/VirtualKey-finalProj--java>

**Unique Selling Points of the Application**

The application keeps running until you exit even if wrong input it asks to give right input.

It was simple and easy to use.

The application takes the folder path and then file name for making it easier.

Navigation from one menu to other is possible.

**Conclusion:**

The Application is prepared according to the **provided flow and features of the application and everything followed according to the instructions.**