PROJECT REPORT

**FILE MANAGEMENT SYSTEM**

Submitted in partial fulfillment of the requirement for the award of degree

OF

**FULL STACK JAVA DEVELOPER**

**PHASE-1 SESSION (5.09.2020-3.10.2020**)

**UNDER THE GUIDENCE OF**

**MR. VINAY**

**COURSE INSTRUCTOR (SIMPLILEARN)**

**SUBMITTED BY**

**MANISH KUMAR ARORA**



**AT STUDY CENTRE**

**Learning Management System**

**SIMPLILEARN**

**CONTENTS**

1. Introduction
   1. Project Details
   2. Developer Details
   3. Problem Description
2. Sprint Plan
   1. Number of Sprints
   2. Duration of Sprint
   3. User Stories
   4. Task achieved in each Sprint
3. Algorithm of application
4. Flow Charts of application
5. Implementation
   1. Software and Tools used
   2. Methodology
   3. Coding
6. Output Test - Output Screens
7. Core Concept used in the Project
8. Links to the GitHub repository to track and verify the project
9. Conclusion
   1. Summary
   2. Future Enhancements
   3. Unique Selling Points(USPs)

**INTRODUCTION**

1. **Project Detail:**

The project “File Management System” aims to increase the scope of user’s interactions by developing menu driven options in order to perform operations on files like: sorting, adding, deleting and searching of files with user given inputs via command line.

1. **Developer Detail:**

Manish Kumar Arora

Engineer-First Level Operations

MBA-IT, B.TECH-CSE

1. **Problem Description:**

Prototype of this project must include Java Concept and Data Structures, which is used to develop the application. Also, include the concept such as exceptions, collection and sorting techniques for source code optimization and increased performance. This application must have following basic features:

1. Option for retrieving the files in ascending order.
2. Option to add a user specified file to the application ignoring the case sensitivity feature.
3. Option to delete a user specified file from the application with case sensitivity feature on the file, ensuring the right file deleted from a given directory.
4. Option to search a user specified file from the application with case sensitivity feature on the file name to retrieve the correct file.
5. Option to return to the main context
6. Option to close the application

**SPRINT PLAN**

1. **Number of Sprints: 2**
2. **Duration of each sprint: One Week**
3. **User Stories: 7 user stories.**

Sprint1: 15.09.2020 – 22.09.2020

User Story1:

**Title: USER MENU**

**Priority: 1**

**Estimate: 1 W**

**As an** End user

**I want to** have multiple options

**So that I can** perform file operations like add, delete, sort and search

User Story2:

**Title: SORT FILE**

**Priority: 2**

**Estimate: 1 W**

**As an** End user

**I want to** sort the files in ascending order for a particular directory

**So that I can** check the files in a directory easily

Sprint2: 23.09.2020 – 30.09.2020

User Story3:

**Title: ADD**

**Priority: 1**

**Estimate: 1 D**

**As an** End user

**I want to** enter a new file name

**So that I can** add new file in that directory

User Story4:

**Title: DELETE**

**Priority: 1**

**Estimate: 2 D**

**As an** End user

**I want to** give directory name and file name

**So that I can** delete that exact same file in that directory

User Story5:

**Title: SEARCH**

**Priority: 1**

**Estimate: 2 D**

**As an** End user

**I want to** give directory name and file name

**So that I can** search exact same file in that directory

User Story6:

**Title: RETURN BACK**

**Priority: 1**

**Estimate: 1 D**

**As an** End user

**I want to** return back to main menu

**So that I can** perform another task

User Story7:

**Title: CLOSE**

**Priority: 1**

**Estimate: 1 D**

**As an** End user

**I want to** exit the application

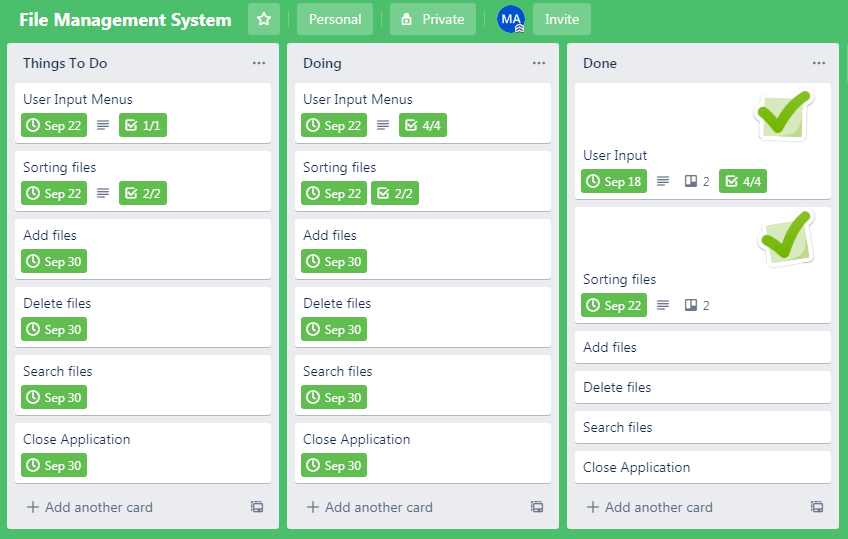
**So that I can** close the application safely.

1. **Task achieved in each sprint:**
2. **Sprint One – Developed Menus for user inputs for all options and develop first business logic option method i.e. to sort files.**
3. **Sprint Two – Developed other business logic option for adding, deleting and searching a user specified file and an extra option to go back to main menu or closing the application.**

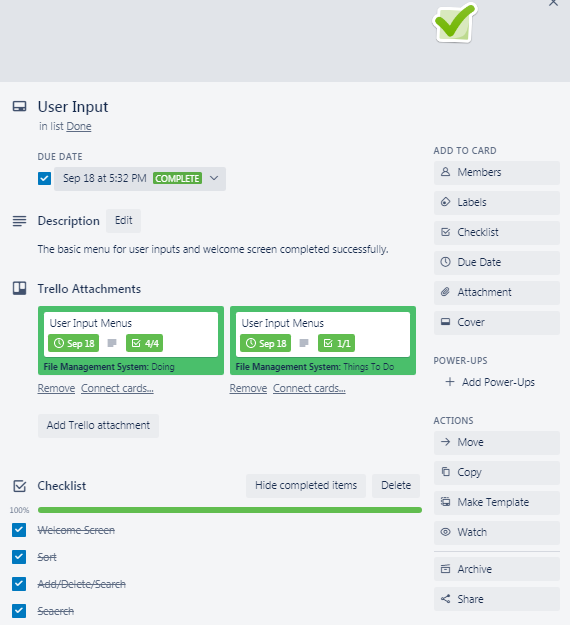
**Detailed Screenshots for each user stories used in scrum planning in each sprint:**

1. **Sprint 1:**

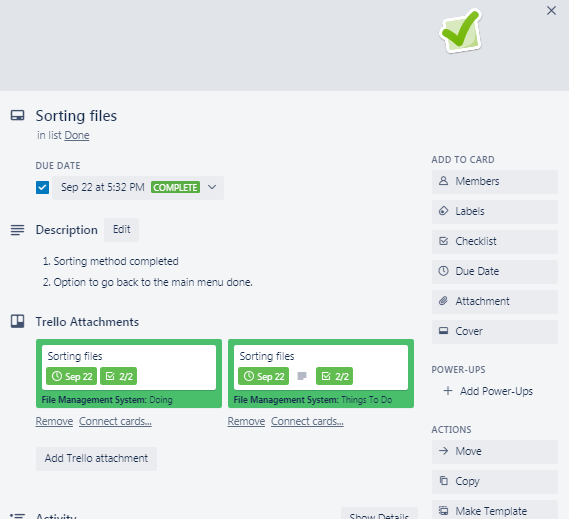
**Date: 15.09.2020 – 22.09.2020**

****

User Input Menus – Completed Date: 18.09.2020

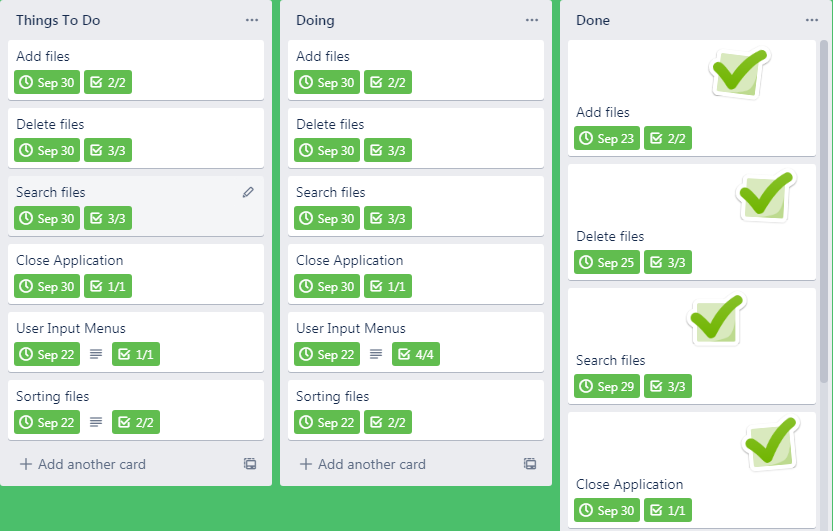


Sorting files – Completed Date: 22.09.2020

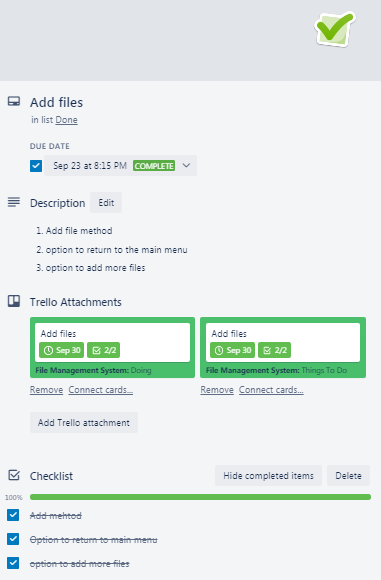


1. **Sprint 2:**

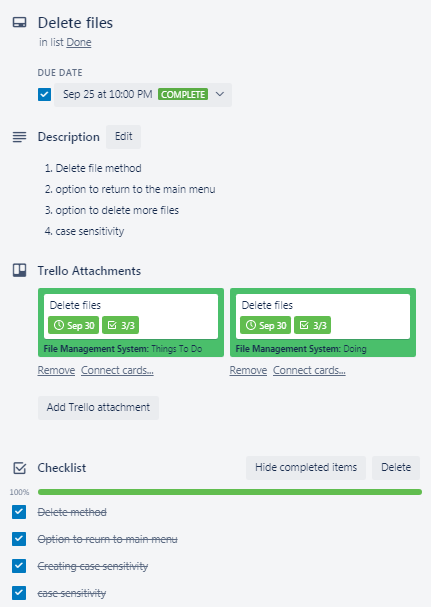
**Date: 23.09.2020 – 30.09.2020**



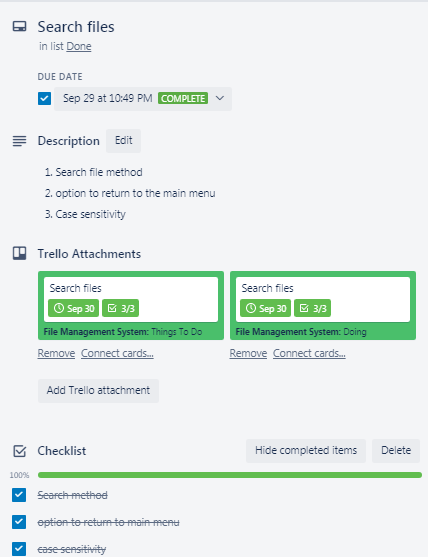
Add files – Completed Date: 23.09.2020



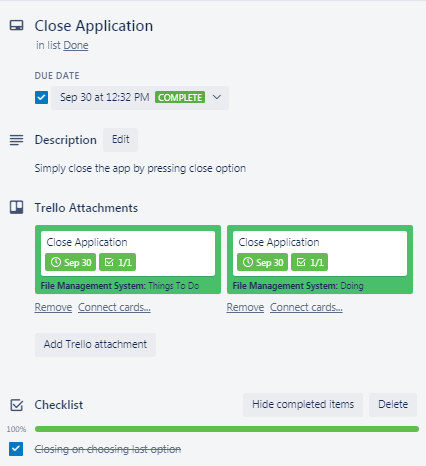
Delete files – Completed Date: 25.09.2020



Search files – Completed Date: 29.09.2020



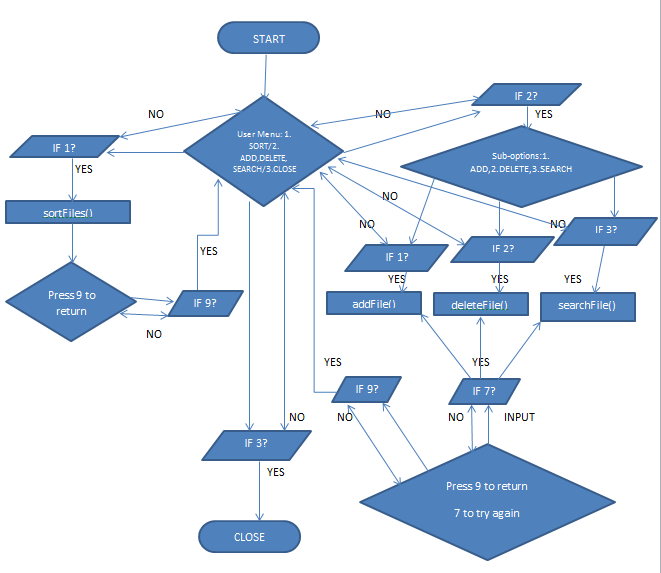
Close Application – Completed Date: 30.09.2020



**ALGORITHM OF APPLICATION**

1. **Class FileMain Algorithm:**
2. Start main()
3. Initialize variable sc to take input
4. Set option = 0, suboption = 0
5. Print Main Menu
6. Do: take input option, till option != 3
7. IF option =1 :: calls-> SortingFiles() :: GO TO B
8. IF option =2 :: Print Sub-Options
9. IF suboption =1:: calls -> AddingFiles() :: GO TO C
10. IF suboption =2:: calls -> DeletingFiles() :: GO TO D
11. IF suboption =3:: calls -> SearchingFile() :: GOT TO E
12. IF option =3:: Exit
13. **Class sortFiles Algorithm:**
14. Start SortingFiles()
15. Initialize String variable afn to take input
16. Create File instance fd
17. Set fd = afn
18. IF fd is Directory:: Create list object l = fd as array list.
19. IF directory is not empty:: calls ->collections.Sort() – sort file list in ascending
20. Iterate for every s = l :: create new file instance f =s
21. IF f is file print files :: GO TO STEP 11
22. Else print no files :: GO TO STEP A.1
23. IF fd is not Directory Print no directory:: GO TO STEP 1
24. Press 9 to return to Main menu
25. Set i = 0
26. Do take input for i, till i!=9
27. IF i=9:: GOTO STEP A.1
28. **Class addFiles Algorithm:**
29. Start AddingFiles()
30. Initialize String variable a and af to take input
31. Create File instance adfile
32. Set adfile = a,af
33. Initialize boolean result = Create new file
34. IF result = true :: Print – File added :: GO TO STEP 9
35. Else :: GO TO STEP 1
36. Else print no files :: GO TO STEP A.1
37. Print - Press 9 to return or 7 to add more
38. Set i = 0
39. Do take input for i, till i!=9
40. IF i=9:: GOTO STEP A.1
41. IF i=7:: GOTO STEP 1
42. **Class deleteFiles Algorithm**:
43. Start DeletingFiles()
44. Initialize String variable dir to take input for directory
45. Create File instance dp
46. Set dp = dir
47. Initialize string array contents = dp as a list
48. Set i=0
49. Print contents till i<length of contents
50. Initialize String variable name to take input for file
51. Set i=0
52. While i<length of contents :: GOTO STEP 11
53. IF contents[i] equals to name :: Create File instance f = dir,name
54. IF f deleted :: Print File Deleted :: GO TO STEP 14
55. Else FILE NOT FOUND :: GO TO STEP 14
56. Print - Press 9 to return or 7 to add more
57. Set i = 0
58. Do take input for i, till i!=9
59. IF i=9:: GOTO STEP A.1
60. IF i=7:: GOTO STEP 1
61. **Class searchFiles Algorithm:**
62. Start SearchingFiles()
63. Initialize String variable dir to take input
64. Create File instance dp
65. Set adfile = dir
66. Initialize string array contents = dp as a list
67. Set i=0
68. Print contents till i<length of contents
69. Initialize String variable searchfile to take input
70. Set i=0
71. While i<length of contents :: GOTO STEP 11
72. IF contents[i] equals to name :: Create File instance f = dir,searchfile::GOTO STEP 12
73. IF f EXIST :: Print File Found :: GO TO STEP 14
74. Else FILE NOT FOUND :: GO TO STEP 14
75. Print - Press 9 to return or 7 to add more
76. Set i = 0
77. Do take input for i, till i!=9
78. IF i=9:: GOTO STEP A.1
79. IF i=7:: GOTO STEP 1

**FLOWCHART OF APPLICATION**



**IMPLEMENTATION**

1. **Software and Tools used:**
2. Operating System - Windows 7 or upper version
3. Eclipse Oxy- To write Java code for application
4. Trello – Online web app for sprint planning and make user cards.
5. GitHub – Online repository to track and submit the project
6. MS-Word – To create user stories and flow chart
7. **Methodology: Agile Scrum**
8. **Coding:**

**Class – FileMain**

**package** fileopeations;

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** FileMain

{

**private** **static** Scanner *sc*;

**public** **static** **void** main(String[] args) **throws** IOException, FileNotFoundException, NumberFormatException

{

*sc* = **new** Scanner(System.***in***); // scan the input

**int** option = 0;

**int** suboption = 0;

//creating class objects

sortFiles sf = **new** sortFiles();

addFile af = **new** addFile();

deleteFile df = **new** deleteFile();

searchFile sef = **new** searchFile();

//Welcome Screen and Menu

System.***out***.println("©2020 - Virtual Key Repositories-(VKR) by Manish Kumar Arora\n");

System.***out***.println("---Main Menu---");

System.***out***.println("1. Sort the files in asscending order");

System.***out***.println("2. Add, Delete or Search a file");

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

System.***out***.println("Choose your option between 1 to 3 and press enter: ");

/\*do while loop includes switch statement cases runs till you didn't press option 3 and option 2 includes further sub-switch statements \*/

**do**

{

**try**

{

option = Integer.*parseInt*(*sc*.nextLine()); // take input from user

}

**catch**(NumberFormatException e)

{

}

**switch**(option) // pass the user specified input

{

**case** 1:

System.***out***.println("You choose an option 1 to sort files.");

sf.SortingFiles(); // calls the method to sort files

**break**;

**case** 2:

System.***out***.println("A. Press 1 to Add a file");

System.***out***.println("B. Press 2 to Delete a file");

System.***out***.println("C. Press 3 to Search a file");

suboption = Integer.*parseInt*(*sc*.nextLine());

**switch**(suboption)

{

**case** 1:

System.***out***.println("You choose option 1 to add a file");

af.AddingFiles(); // calls the method to add a file

**break**;

**case** 2:

System.***out***.println("You choose option 2 to delete a file");

df.DeletingFiles(); // calls the method to delete files

**break**;

**case** 3:

System.***out***.println("You choose option 3 to search a file");

sef.SearchingFile(); // calls the method to search files

**break**;

}

**break**;

**case** 3:

System.***out***.println("Thank you for using VKR. Bye-Bye..");

**break**;

**default**:

System.***out***.println("Invalid Option! Choose only number between 1 to 3");

**break**;

}

}

**while**(option!=3);

}

}

**Class – sortFiles**

**package** fileopeations;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.\*;

**public** **class** sortFiles **throws** IOException, NumberFormatException

{

Scanner sc = **new** Scanner(System.***in***); // scan user input

**public** **void** SortingFiles()

{

System.***out***.println("Enter your directory path where you want to sort the files - ");

String afn = sc.nextLine(); // take user input

File fd = **new** File(afn); // create new file instance

// File fd = new File("D:\\eclipse-Oxy\\Simplilearn\\Phase1\_Project1\_FileSADS\\FileFolder");

**if**(fd.isDirectory()) // check if the given input is a directory or not

{

List<String> l = Arrays.*asList*(fd.list()); // Returns an array of strings as a list

/\*if condition to check any file present or not if yes then list of files pass to collection sort method to sort the files in ascending order\*/

**if**(fd.list().length>0)

{

Collections.*sort*(l);

System.***out***.println("Sorting the files in ascending Order...\n");

**for**(String s:l) // iterate the list

{

File f = **new** File(s); // creating new file instance

**if**(f.isFile()) // check only for files

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

}

}

**else**

{

System.***out***.println(fd.getName() + " This folder is empty...");

System.***out***.println("Please choose option 2 and add some files.\n");

FileMain.*main*(**null**);

}

}

**else**

{

System.***out***.println(fd.getAbsolutePath() + " is not a directory");

System.***out***.println("Please Enter correct path and try again..\n");

SortingFiles();

}

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

System.***out***.println("Press 9 to return to the main menu: ");

**int** i=0;

/\*do while loop to return to the main menu will continue to run till 9 not pressed\*/

**do**

{

**try**

{

i = Integer.*parseInt*(sc.nextLine());

**if**(i==9)

{

FileMain.*main*(**null**);

}

**else**

{

System.***out***.println("Wrong Input! " +i +" is not a valid number. Press 9 to return-");

}

}

**catch**(NumberFormatException e)

{

System.***out***.println(e.getMessage()+ " is not a number. Press number 9 and enter to return-");

}

}

**while**(i!=9);

}

}

**Class – addFiles**

**package** fileopeations;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** addFile **throws** IOException, NumberFormatException

{

Scanner sc = **new** Scanner(System.***in***); // scan user input

**public** **void** AddingFiles()

{

System.***out***.println("Enter directory path:");

String a = sc.nextLine(); // take user input for directory path

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

String af = sc.nextLine(); // take user input for adding a file

File adfile = **new** File(a,af);

// File adfile = new File("D:\\eclipse-Oxy\\Simplilearn\\Phase1\_Project1\_FileSADS\\FileFolder");

**boolean** result;

**try**

{

result = adfile.createNewFile(); // create a new file

**if**(result) // test if successfully created a new file

{

System.***out***.println("New File added: " + adfile.getCanonicalPath());

}

**else**

{

System.***out***.println("File with same name already exist at a location: " + adfile.getCanonicalPath());

System.***out***.println("Please try again.. ");

AddingFiles();

}

}

**catch**(IOException e)

{

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

}

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

System.***out***.println("A. Press 9 to return to the main menu");

System.***out***.println("B. Press 7 to try again or add more files");

**int** i=0;

/\*do while loop to return to the main menu or adding more files. Loop will continue to run till 9 not pressed\*/

**do**

{

**try**

{

i = Integer.*parseInt*(sc.nextLine());

**if**(i==9)

{

FileMain.*main*(**null**);

}

**else** **if**(i==7)

{

AddingFiles();

}

**else**

{

System.***out***.println("Wrong Input! " +i +" is not a valid number. Press 9 to return or 7 to add more files-");

}

}

**catch**(NumberFormatException e)

{

System.***out***.println(e.getMessage()+ " is not a number. Press number 9 and enter to return or 7 to add more files-");

}

}

**while**(i!=9);

}

}

**Class – deleteFile**

**package** fileopeations;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** deleteFile **throws** IOException, FileNotFoundException

{

//Scanner d = new Scanner(System.in); //

**private** Scanner sc;

**public** **void** DeletingFiles()

{

sc = **new** Scanner(System.***in***); // take user input for directory path

System.***out***.println("Enter directory path: ");

String dir = sc.nextLine();

File dp = **new** File(dir); // creating new file instance passing user input

String contents[] = dp.list(); //creating the list and passing it to array contents

System.***out***.println("List of files in the specified directory:");

**for**(**int** i=0; i<contents.length; i++) //iterate the array list and print the list of files

{

System.***out***.println(contents[i]);

}

System.***out***.println("\nEnter the name of file you want to delete: ");

String name = sc.nextLine(); //take user input for file name to delete

**int** i=0;

**try**

{

**while**(i<=contents.length) //loop runs until all file iterate and finds a match

{

/\* check the file name entered matches the file in directory with respect to case sensitivity if yes then delete else break the loop\*/

**if**(contents[i].equals(name))

{

File f = **new** File(dir,name); //creating new file instance and passing path and file name to it.

**if**(f.delete())

{

System.***out***.println(f.getName()+ " deleted successfully");

**break**;

}

}

i++;

}

}

**catch**(Exception e)

{

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

}

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

System.***out***.println("A. Press 9 to return to the main menu");

System.***out***.println("B. Press 7 to try again or delete more files");

**int** k=0;

/\*do while loop to return to the main menu or deleting more files. Loop will continue to run till 9 not pressed\*/

**do**

{

**try**

{

k = Integer.*parseInt*(sc.nextLine());

**if**(k==9)

{

FileMain.*main*(**null**);

}

**else** **if**(k==7)

{

DeletingFiles();

}

**else**

{

System.***out***.println("Wrong Input! " +k +" is not a valid number. Press 9 to return-");

}

}

**catch**(NumberFormatException e)

{

System.***out***.println(e.getMessage()+ " is not a number. Press number 9 and enter to return-");

}

}

**while**(k!=9);

}

}

**Class – searchFile**

package fileopeations;

import java.io.File;

import java.io.IOException;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class searchFile throws IOException, FileNotFoundException

{

private Scanner sc;

public void SearchingFile()

{

sc = new Scanner(System.in); // scan the input

System.out.println("Enter directory path: ");

String dir = sc.nextLine(); // take user input for directory path

File dp = new File(dir); // creates new file instance and passing user given input

String contents[] = dp.list(); //creating the list and passing it to array contents

System.out.println("List of files in the specified directory:");

for(int i=0; i<contents.length; i++) //iterate the array list and print the list of files

{

System.out.println(contents[i]);

}

System.out.println("Enter File name to search:");

String searchfile = sc.nextLine(); //take user input for file name to search

int i=0;

try

{

while(i<=contents.length) //loop runs until all file iterate and finds a match

{

/\* check the file name entered matches the file in directory with respect to case sensitivity if yes then print the location and break the loop\*/

if(contents[i].equals(searchfile))

{

File f = new File(dir,searchfile);

if(f.isFile()&&f.exists())

{

System.out.println(f.getName()+ " found at location: " +f.getAbsolutePath());

break;

}

}

i++;

}

}

catch(Exception e)

{

System.out.println("FILE NOT FOUND");

}

System.out.println();

System.out.println("A. Press 9 to return to the main menu");

System.out.println("B. Press 7 to try again or search more files");

int k=0;

/\*do while loop to return to the main menu or searching more files. Loop will continue to run till 9 not pressed\*/

do

{

try

{

k = Integer.parseInt(sc.nextLine());

if(k==9)

{

FileMain.main(null);

}

else if(k==7)

{

SearchingFile();

}

else

{

System.out.println("Wrong Input! " +k +" is not a valid number. Press 9 to return to main menu or 7 to search more-");

}

}

catch(NumberFormatException e)

{

System.out.println(e.getMessage()+ " is not a number. Press number 9 and return to main menu or 7 to search more-");

}

}

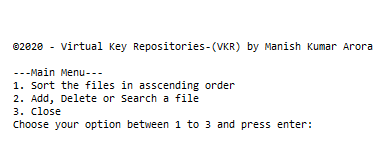
while(k!=9);

}

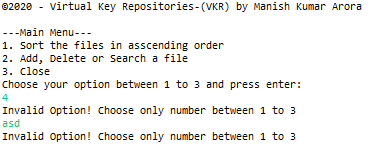
}

**OUTPUT TEST – OUTPUT SCREENSHOTS**

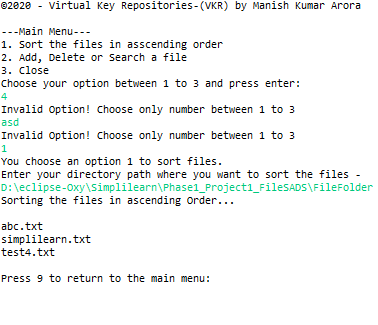
1. **Welcome Screen:**

****

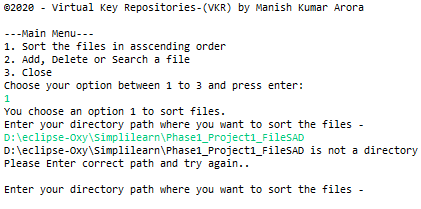
1. **Invalid Input:**

****

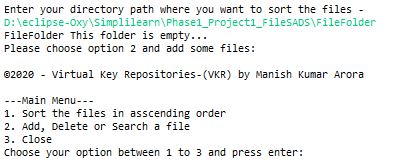
1. **Sorting Files:**

****

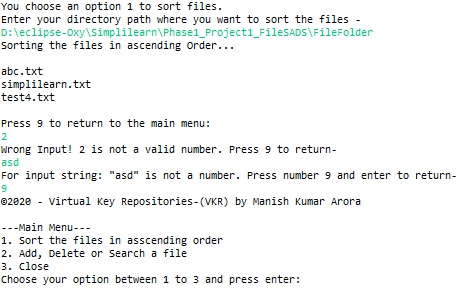
1. **Checking if directory path entered is correct in sorting method:**

****

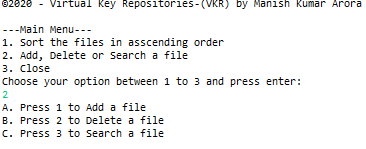
1. **Checking if folder is empty in sorting method:**

****

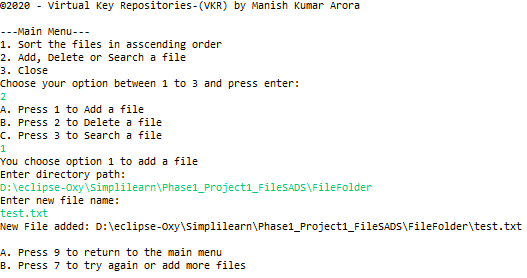
1. **Return to main from sorting files:**

****

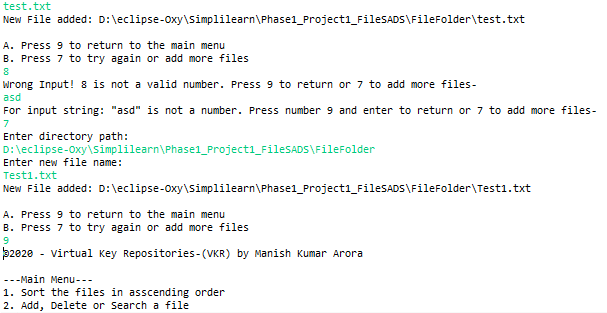
1. **Showing sub-options when press 2:**

****

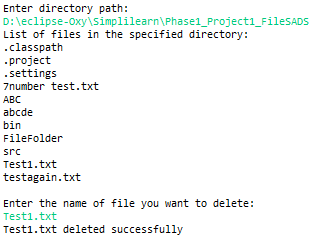
1. **Adding Files:**

****

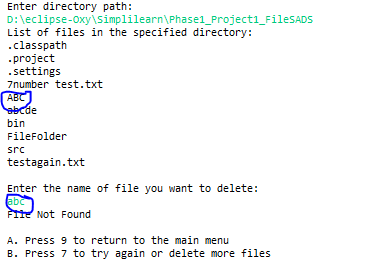
1. **Adding more files by pressing 7 and 9 to return to main menu:**

****

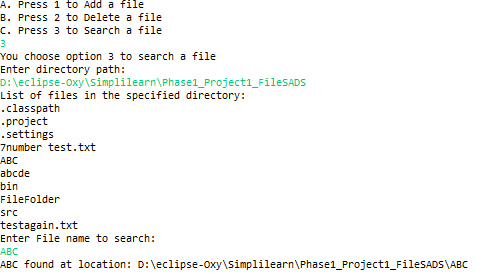
1. **Deleting Files:**

****

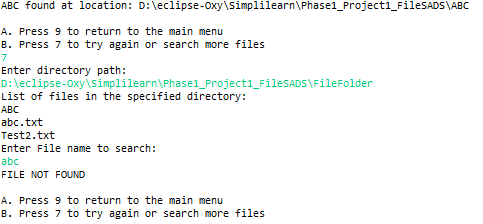
1. **Checking case sensitivity on deletion:**

****

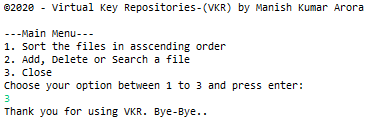
1. **Searching Files:**

****

1. **Checking case sensitivity in searching:**

****

1. **Exit the application:**

****

**CORE CONCEPT USED IN THE PROJECT**

This project uses Java Programming language features. Brief of all concepts used in the project are as follow:

1. Scanner class for input/output operation.
2. File class to handle file operations.
3. Collections framework for listing the files as array list and sorting the list using collections’ sort method.
4. If and Switch statements for checking the conditions.
5. For and do-while loops for iterating through the list of files in a directory.
6. Exception Handling features to catch errors if occurred.

**LINK TO THE GITHUB REPOSITORY TO TRACK AND VERIFY THE PROJECT COMPLETION**

<https://github.com/Manish-K-Arora/SimpliLearnPhase1Project.git>

**CONCLUSION**

1. **Summary**

* The project has been developed mainly for file management purpose.
* It is easy to use, since it uses console inputs from users via command line
* User friendly Menu Driven options are provided to user with variable options for performing file operations
* The usage of software increases the efficiency, decreases the effort.
* It has been thoroughly tested and implemented.

1. **Future Enhancements**

Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* We can also add additional features to write in file during creation or transfer the contents of another file to newly added files.
* Based on the future security issues, security can be improved using the techniques to enter user name or password to use the application.

1. **Unique Selling Points(USPs)**

This is a menu driven easy to use and perform file operations through user input via command line interface which enhances the interaction of user with application and includes case sensitive features in deletion and searching of file in order to ensure the correct file deleted or retrieved.