**Locked Me – Virtual Key for Repositories**

This document contains sections for:

1. [Project Details and Problem Statement](#DESCRIPTION)
2. [Sprint planning and Task completion](#Sprint_plan)
3. [Algorithm and flowchart](#Algorithm)
4. Source Code
5. Output Screen Shot
6. Conclusion

The code for this project is hosted at <https://github.com/amsdhs36/VirtualRepository>

The project is developed by Amruta Singh.

**1.Project 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.

**Background of the problem statement:**

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You’re asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you’re asked to present the following in the next 15 working days (3 weeks):

* Specification document - Product’s capabilities, appearance, and user interactions
* Number and duration of sprints required
* Setting up Git and GitHub account to store and track your enhancements of the prototype
* Java concepts being used in the project
* Data Structures where sorting and searching techniques are used.
* Generic features and three operations:
  + Retrieving the file names in an ascending order
  + Business-level operations:
    - Option to add a user specified file to the application
    - Option to delete a user specified file from the application
    - Option to search a user specified file from the application
    - Navigation option to close the current execution context and return to the main context
  + Option to close the application

The goal of the company is to deliver a high-end quality product as early as possible.

## **2. Project planning and Task completion**

* Initializing git repository to track changes as development progresses.
* Prepare flowchart and algorithm
* Writing the Java program to fulfill the requirements of the project.
* Pushing code to GitHub.

The project is planned to be completed in 2 sprints. Tasks assumed to be completed in the sprint are:

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | User Story | Tasks | Duration |
| 1 | As a user, I need to see the application welcome screen with Menu options. Menu First Option: User should able to display the files in ascending order from the directory. The root directory can be either empty or contain few files or folders in it.Second Option Exit | Create the Welcome screen | 1 |
| Create Check if directory exist or create new | 1 |
| If exist and choice enter by user to display the files in ascending order | 1 |
| 2 | Second option: Another menu inside Second option with choices -User should be able to 1.add, 2. delete files, 3. search file from the directory.  4.User should be able to get back to previous menu  Third Option: Exit the menu | Add methods to add the file in the directory, delete method to delete file | 3 |
| Search the user specified file from directory and give error message if not found | 2 |
| Go back to previous menu or exit the menu | 1 |

**3.Algorithm and Flowchart**

**User Story1**

**Steps:**

1. Start
2. Check if directory exist else create a new directory in the project
3. Print the welcome screen
4. Display the menu
5. Take user choice from menu and switch to user choice
6. If user choice 1 then display the files inside directory in ascending order
7. If user choice 2 then exit
8. Default statement: Please enter correct input

**Algorithm to display the files inside the directory in ascending order**

**Steps:**

1. Create a File object for the main directory.
2. Get an array of files for the main directory.
3. If array[i] is a directory: Print out directory name
4. If array[i] is a file: Print out the file name.(recursively)

**User Story2**

**Steps:**

1. Start
2. Check if directory exist else create a new directory in the project
3. Print the welcome screen
4. Display the menu
5. Take user choice from menu and switch to user choice
6. If user choice 1 then display the files inside directory in ascending order
7. If user choice 2 then display the second menu
8. 1.Add: If user choice 1 then add the file as specified by user
9. 2. Delete: If user choice 2 then delete the file as specified by user
10. 3. Search: If user choice 3 then search the file as specified by user
11. 4. If user choice 4 then go to step 4
12. If user choice 5 then exit
13. Default statement: Please enter correct input
14. If user choice 3 then exit
15. Default statement: Please enter correct input

**Flowchart:**

:

Start

Create “Main” folder if not exist

Take Input and Switch

Take Input and Switch

Print Welcome Screen and app details

Case 1se 1

Add the file and display submenu

Case 1se 1

Display all files from the Main folder and then display menu again

Display Menu

1. Display the files
2. Display the sub menu
3. Exit

**Case 2** 222

Display Sub menu:

1.Add file 2. Delete File 3.Search File

4.Goto main menu 5.Exit

Case 2 222

Delete the file and display submenu

Search the file and display submenu

**Case 3** 3

**Case 3** 3

**Case 4** 3

Go to main menu

Default : Display message Please enter correct input.

**Case 5** 3

End

Default : Display message Please enter correct input.

**Source Code:**

Program: MainLockedMe.java

**package** LockedMe;

**public** **class** MainLockedMe {

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

// **TODO** Auto-generated method stub

FileOperations.*directoryifnotexist*("Main");

WelcomeScreenMenu.*WelcomeScreen*("LockeMe-VirtualRepository", "Amruta");

UserChoice.*gotoUserChoice*();

}}

Program: FileOperation.java

**package LockedMe;**

**import java.io.File;**

**//import java.io.FilenameFilter;**

**import java.io.IOException;**

**import java.nio.file.Files;**

**import java.nio.file.Path**

**import java.nio.file.Paths;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.List;**

**import java.util.Scanner;**

**public class FileOperations {**

**public static void directoryifnotexist(String folderName) {**

**File file = new File(folderName);**

**// If directory not exist, create the Main folder**

**if (!file.exists()) {**

**file.mkdirs();**

**}**

**}**

**public static void displayAllFiles(String path) {**

**FileOperations.directoryifnotexist("Main");**

**// All required files and folders inside "main" folder**

**System.out.println("Displaying all files form the directory structure\n");**

**// listFilesInDirectory displays files along with folder structureList<String> filesListNames = FileOperations.listDirectoryFiles(path, 0, new ArrayList<String>());**

**System.out.println("Displaying all files in ascending order\n");**

**Collections.sort(filesListNames);**

**filesListNames.stream().forEach(System.out::println);**

**}**

**public static List<String> listDirectoryFiles(String dirName,int Count,List<String> fileListNames)**

**{**

**FileOperations.directoryifnotexist("Main");**

**File dir = new File(dirName);**

**File[] files = dir.listFiles();**

**List<File> filesList = Arrays.asList(files);**

**Collections.sort(filesList);**

**if (files != null && files.length > 0) {**

**for (File file : filesList) {**

**System.out.print(" ".repeat(Count \* 2));**

**if (file.isDirectory()) {**

**System.out.println("`-- " + file.getName());**

**// Recursively indent and display the files**

**fileListNames.add(file.getName());**

**listDirectoryFiles(file.getAbsolutePath(), Count + 1, fileListNames);**

**} else {**

**System.out.println("|-- " + file.getName());**

**fileListNames.add(file.getName());**

**}**

**}**

**} else {**

**System.out.print(" ".repeat(Count \* 2));**

**System.out.println("|-- Empty Directory");**

**}**

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

**return fileListNames;**

**}**

**public static void addFile(String fileToAdd, Scanner sc) {**

**FileOperations.directoryifnotexist("Main");**

**Path FilePath = Paths.get("./main/" + fileToAdd);**

**try {**

**Files.createDirectories(FilePath.getParent());**

**Files.createFile(FilePath);**

**System.out.println(fileToAdd + " successfully created ");**

**} catch (IOException e) {**

**System.out.println("Failed,file not created " + fileToAdd);**

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

**}**

**}**

**public static void deleteFile(String fileToDel)**

**{**

**FileOperations.directoryifnotexist("Main");**

**Path Filepath=Paths.get("./Main/"+fileToDel);**

**try**

**{**

**Files.delete(Filepath);**

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

**}**

**catch (IOException e)**

**{**

**System.out.println("Failed, filed not deleted"+ fileToDel);**

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

**}**

**}**

**public static void searchFile(String fileToSearch)**

**{**

**File directory=new File("Main/");**

**String[] fileList=directory.list();**

**int flag=0;**

**if(fileList==null)**

**{**

**System.out.println("Empty Directory");**

**}**

**else**

**for(int i=0;i<fileList.length;i++)**

**{**

**String filename=fileList[i];**

**if(filename.equalsIgnoreCase(fileToSearch))**

**{**

**System.out.println(filename+" file found in main folder");**

**flag=1;**

**Path currentRelativePath=Paths.get(directory.getAbsolutePath());**

**String s=currentRelativePath.toAbsolutePath().toString();**

**System.out.println("main is located at "+s);**

**}**

**}**

**if(flag==0)**

**{**

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

**}**

**}**

**}**

**Program: UserChoice.java**

**package** LockedMe;

**import** java.util.Scanner;

**public** **class** UserChoice {

**public** **static** **void** gotoUserChoice() {

**boolean** running = **true**;

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

**do** {

**try** {

WelcomeScreenMenu.*displayMenu*();

System.***out***.println("Please enter your choice:");

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

**switch** (input) {

**case** 1:

FileOperations.*displayAllFiles*("Main");

**break**;

**case** 2:

UserChoice.*gotoUserSubMenuChoice*();

**break**;

**case** 3:

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

running = **false**;

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please enter valid choice from the menu");

}

}**catch** (Exception e) {

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

*gotoUserChoice*();

}

} **while** (running == **true**);

}

**public** **static** **void** gotoUserSubMenuChoice() {

**boolean** running=**true**;

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

**do** {

**try** {

WelcomeScreenMenu.*displaySubMenu*();

System.***out***.println("Please enter your choice:");

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

**switch**(userinput) {

**case** 1:

System.***out***.println("Enter the name of the file to be added to the Main folder");

String fileToAdd = sc.next();

FileOperations.*addFile*(fileToAdd, sc);

**break**;

**case** 2:

System.***out***.println("Enter the name of the file to be deleted");

String fileToDel=sc.next();

FileOperations.*deleteFile*(fileToDel);

**break**;

**case** 3:

System.***out***.println("Enter the file to be searched");

String fileToSearch=sc.next();

FileOperations.*searchFile*(fileToSearch);

**break**;

**case** 4:

**return**;

**case** 5:

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

running = **false**;

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please enter valid choice from the Sub-menu");

}

}**catch** (Exception e) {

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

*gotoUserSubMenuChoice*();

}

}**while** (running==**true**);

}

Program: WelcomeScreenMenu.java

**package** LockedMe;

**public** **class** WelcomeScreenMenu {

//Display application name and developer name

**public** **static** **void** WelcomeScreen(String appName, String developerName) {String Details = String.*format*("\* Welcome to world of %s.com. \n" + "\*Developed by %s .\n",

appName, developerName);

String appFunction = "Hello User ,you can use this application to\n"

+ "Retrieve all file names in the \"Main\" folder\n";

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

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

}

//Display the menu

**public** **static** **void** displayMenu() {

String menu = "\n\nSelect any option number from below and press Enter \n\n"

+ "1) Retrieve all files inside \"main\" folder\n"

+ "2) Display the File SubMenu\n"

+ "3) Exit program\n";

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

}

//Display the sub menu: add, delete,search file,return to previous menu

**public** **static** **void** displaySubMenu() {

String menu2="\n\nSelect any option from Submenu and press Enter\n\n"

+ "1. Add the file\n"

+ "2. Delete the file\n"

+ "3. Search the file\n"

+ "4. Return to the main menu\n"

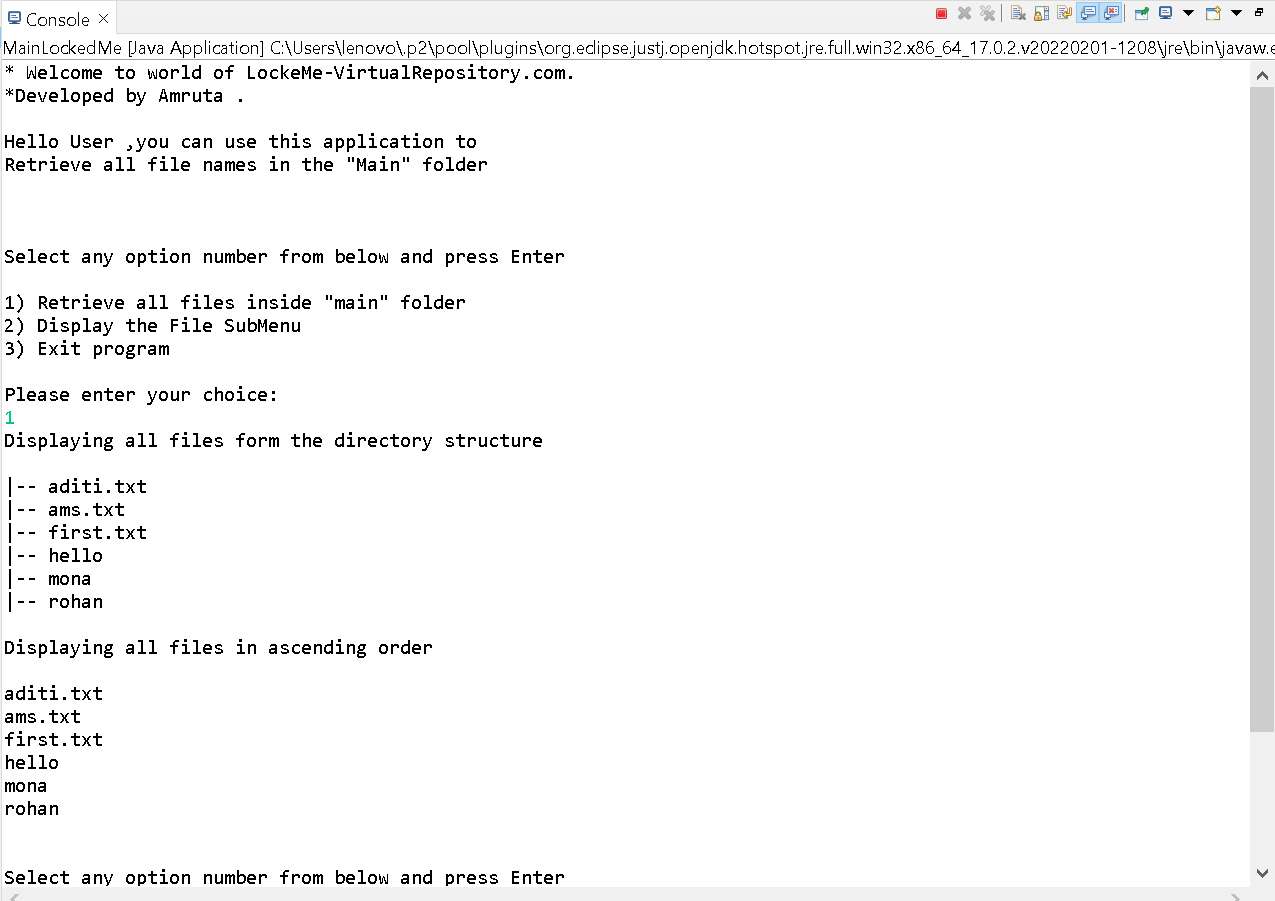
+ "5. Exit program\n";

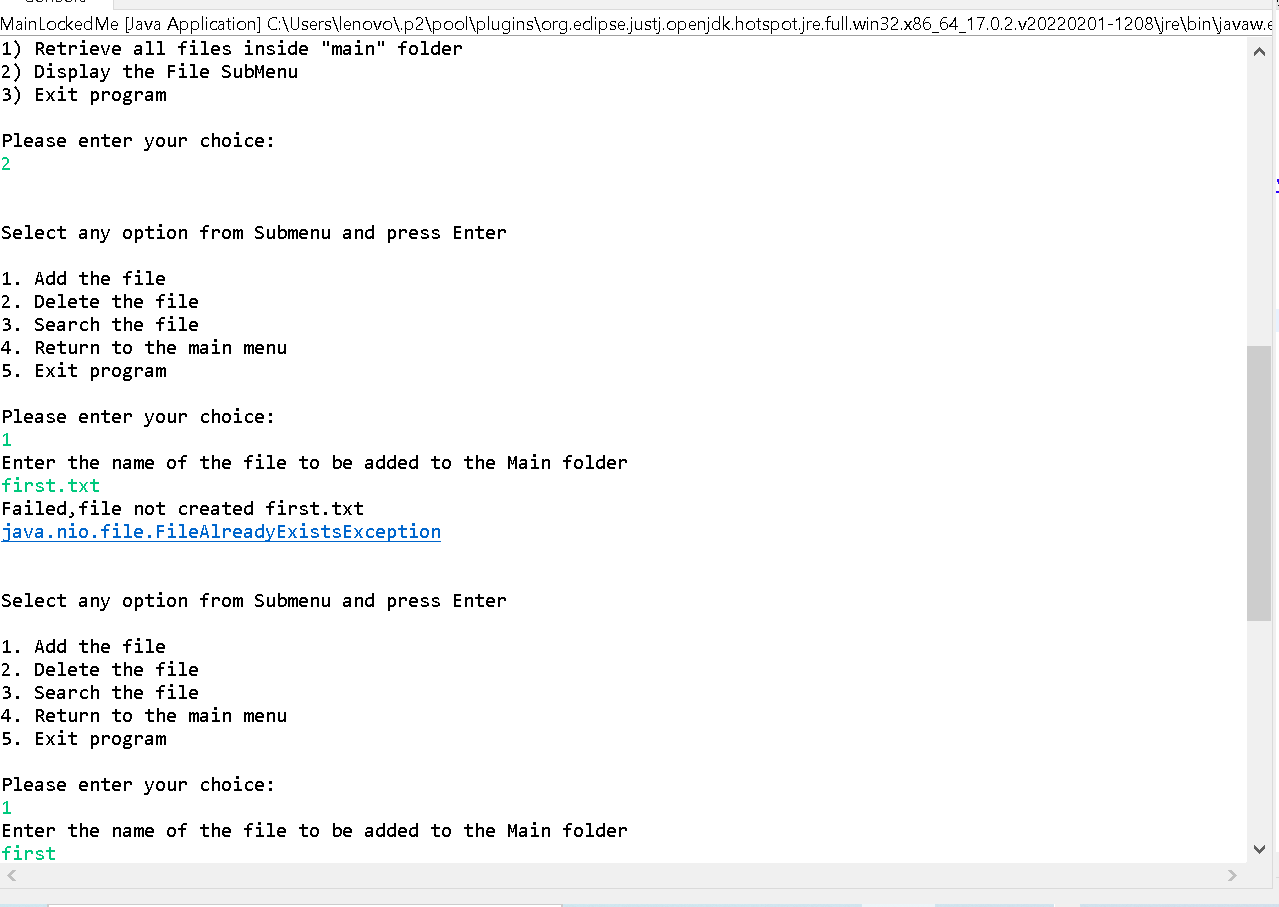
System.***out***.println(menu2);

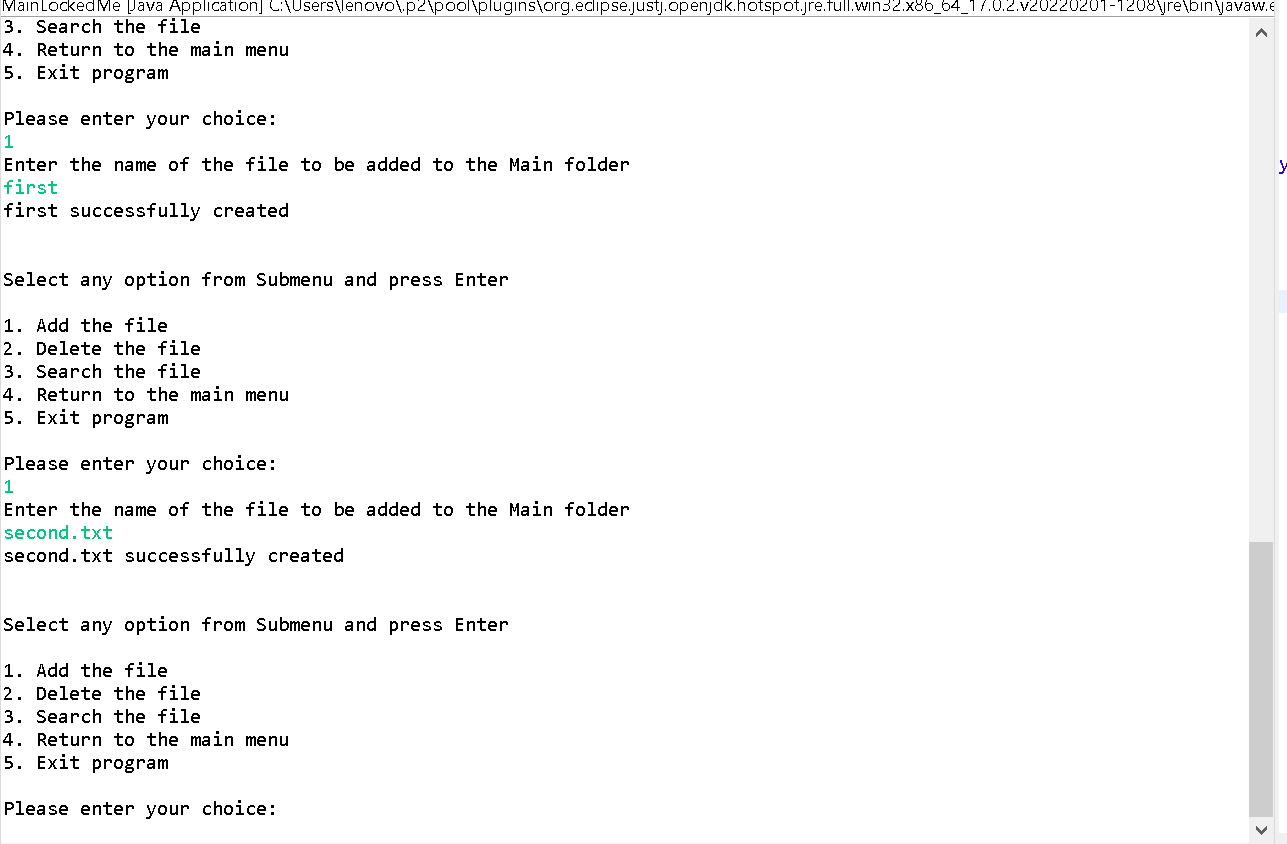
}

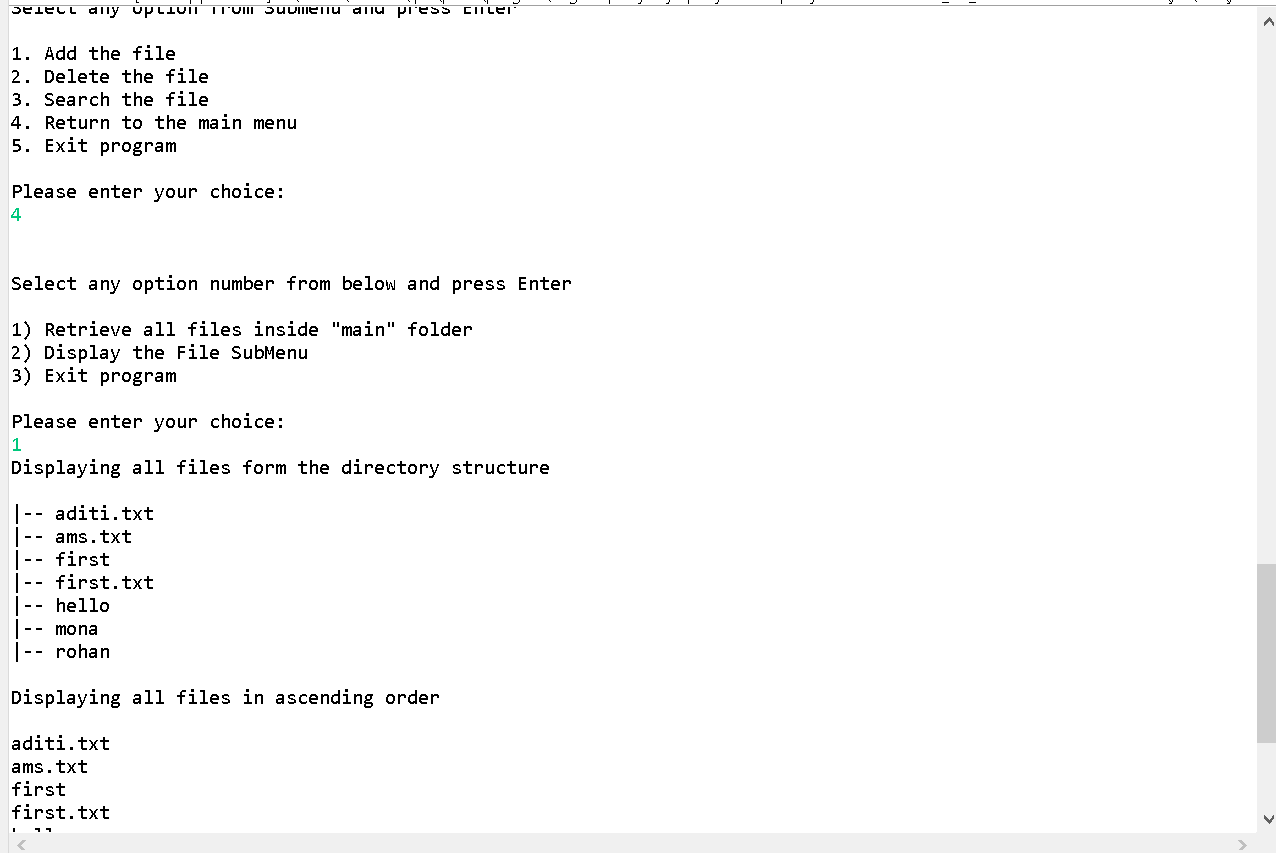
}

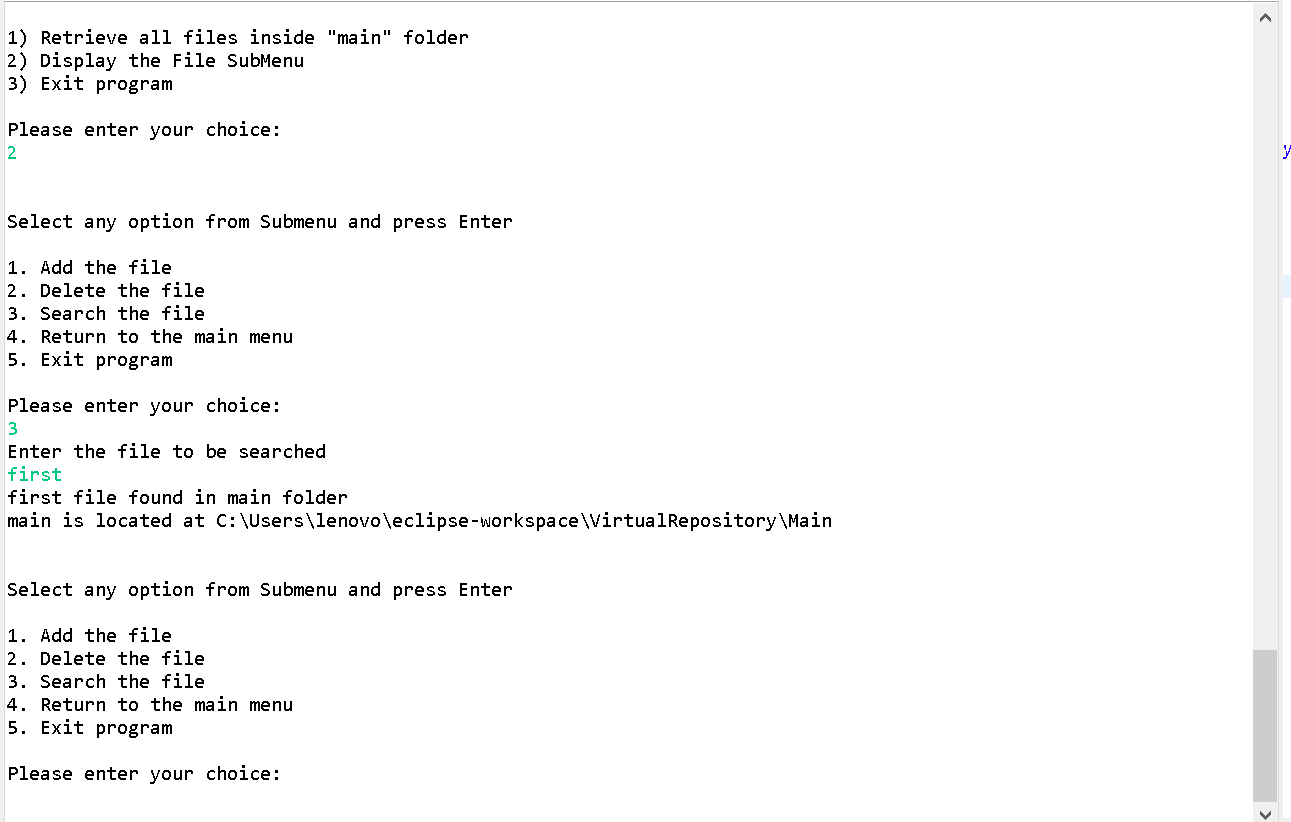
**Output:**

****

****

****

****

****