LockedMe – Virtual Key for Repositories

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

The project is developed by Dinesh Reddy

**Sprints planning and Task completion**

Project work divided into 2 Sprint 10 days, Sprint work details are –

**1st – Sprint Work Details -**

* Initializing git repository to track changes as development progresses.
* Create Java class to show the files from the folder in ascending order.
* Create Java class for Option to add a user specified file to the application.
* Create Java class for Option to search a user specified file from the application with case

sensitive.

* Create Java class for Option to read a user specified file from the application with case

sensitive.

* Push the Java class in the git repository.

**2nd – Sprint Work Details –**

* Create Java class for Option to delete user specified file from the application with case

sensitive.

* Create Main Java class as a welcome class to Navigation option to close the current

execution context and return to the main context.

* Welcome Class is the main class to execute all the classes.
* Test Again All the Java class.
* Migrate all the Java classes to the welcome class and execute the application.
* Test applications make the required changes.
* Push the Java class in the git repository.

**Core concepts used in project**

Collections framework, File Handling, Sorting, Flow Control, Exception Handling, Loop, Algorithm, sorting etc.

**Java Classes to use in project –**

* **WelcomeScreen Class -** Used to welcome the user, Main menu and File operation menu.
* **ShowFile Class -** Used to Show Files names in an ascending order present in the directory.
* **AddFile Class -** Used to create a new file, input taken from user for filename and some write up in file.
* **SearchFile Class** - Search a user specified file from the main directory.
* **DeleteFile Class** -Delete a user specified file from the existing directory list.
* **WriteFile Class** - Write the content of the file from the existing directory list.

**Details Concept for Each class with code and Result –**

**1.WelcomeScreen Class -**

* First this class displays the welcome screen and then 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.

i.Add a file to the existing directory list

ii.Delete a user specified file from the existing directory list

iii.Search a user specified file from the main directory

iv. ShowAllFile to Files names in an ascending order present in the

v.Option to navigate back to the main context.

* After Performing operation it will return to the main menu if you want to continue or exit .
* Don’t want to continue type e and it will exit the application .

**1)Code for WelcomeScreen Class**

**package** userInterface;

**import** java.util.Scanner;

**import** operations.AddOperation;

**import** operations.DeleteFile;

**import** operations.SearchFile;

**import** operations.ShowAllFiles;

**import** operations.WriteFile;

**public** **class** WelcomeScreen {

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

System.***out***.println("Welcome to LockedMe Application");

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

System.***out***.println(" Developed by : Dinesh Reddy");

System.***out***.println("Email Id:singireddydinesh1@gmail.com");

**while**(**true**) {

System.***out***.println("Enter your Choice ");

System.***out***.println("1. Show Files names in an ascending order present in the directory");

System.***out***.println("2. File Menu options");

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

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

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

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

**if**(choice==1) {

ShowAllFiles sf = **new** ShowAllFiles();

sf.show();

}

**else** **if**(choice==2) {

**while**(**true**) {

System.***out***.println("a : Add a new File");

System.***out***.println("b : write in a new File");

System.***out***.println("c : Delete a file");

System.***out***.println("d : Search a file");

System.***out***.println("e : Exit (from file menu)");

String choice1=input.next();

**if**(choice1.equals("e")) {

System.***out***.println(" Moved to the main menu ");

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

**break**;

}

**switch**(choice1) {

**case** "a":

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

AddOperation af=**new** AddOperation();

af.createFile();

**break**;

**case** "b":

System.***out***.println(" Enter file name to written in the folder ");

WriteFile wf=**new** WriteFile();

wf.writefile();

**break**;

**case** "c":

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

DeleteFile df= **new** DeleteFile();

df.delete();

**break**;

**case** "d":

System.***out***.println(" Enter file name to be seached ");

SearchFile sf=**new** SearchFile();

sf.search();

**break**;

**default**:

System.***out***.println("Please enter a correct choice");

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

}

}

}

**else** **if**(choice==3) {

System.***out***.println(" Thanks for using Lockedme.com services.");

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

**break**;

}

**else** {

System.***out***.println(" Please Enter a Correct Choice ");

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

}

}

}

}

**OutPut Result for WelcomeScreen Class -Welcome to user ,Main menu and FileOperation menu -**





**2)Code for ShowAll Files**

package operations;

import java.io.File;

import java.util.Arrays;

public class ShowAllFiles {

public void show() {

// Creates an array in which we will store the names of files and directories

String[] pathnames;

// Creates a new File instance by converting the given pathname string

// into an abstract pathname

File f = new File("C:\\Users\\singi\\Downloads\\test\\");

// Populates the array with names of files and directories

pathnames = f.list();

Arrays.sort(pathnames);

System.out.println("Files names in an ascending order ");

// For each pathname in the pathnames array

for (String pathname : pathnames) {

// Print the names of files and directories

System.out.println(" |- " + pathname);

}

System.out.println();

}

}

**OutPut Result for ShowAllFile Class -**



**3) Code for Add file**

package operations;

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

public class AddOperation {

public void createFile() {

try {

Scanner filename = new Scanner(System.in);

String filename1 = filename.next();

File file = new File("C:\\Users\\singi\\Downloads\\test\\" + filename1 );

if (file.createNewFile()) {

System.out.println("File created: " + file.getName());

} else {

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

}

} catch (IOException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

}

}

**OutPut Result for AddFile**



**4)Code for WriteFile**

package operations;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class WriteFile {

public void writefile() {

try {

Scanner filename = new Scanner(System.in);

String filename1 = filename.next();

FileWriter myWriter = new FileWriter("C:\\Users\\singi\\Downloads\\test\\"+ filename1);

System.out.println("Enter your message");

Scanner msg=new Scanner(System.in);

String msg1=msg.nextLine();

myWriter.write(msg1);

myWriter.close();

System.out.println("Successfully wrote to the file.");

}

catch (IOException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

}

}

**OutPut Result for WriteFile**



5)Code for DeleteFile

package operations;

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

public class DeleteFile {

public static String FILE\_LOCATION = "C:\\Users\\singi\\Downloads\\test\\";

public void delete() {

Scanner filename = new Scanner(System.in);

String file = filename.next();

File file1= new File("C:\\Users\singi\Downloads\\test\\");

System.out.println("file : "+file1.getAbsolutePath());

if (file1.delete()) {

System.out.println(" Your Have Successfully deleted the file " + file1.getName());

} else {

System.out.println("File Not Found -FNF .");

}

**}**

**}**

**OutPut Result for Delete File**



**6)Code For Search File**

package operations;

import java.io.File;

import java.util.Scanner;

public class SearchFile {

public void search() {

Scanner filename = new Scanner(System.in);

String file = filename.nextLine();

File file1= new File("C:\\Users\\singi\\Downloads\\test\\" +file);

if(file1.exists()) {

System.out.println(file1.getName() + " is found");

}else {

System.out.println("The file does not exist");

}

}

}

**OutPut Result for Search File**



**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

**Unique Selling Points of the Application**

* 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.

* 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.

* User is also provided the option to write content if they want into the newly created

file.

* 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.

* The application also allows users to delete folders which are not empty.
* 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 are performed.

* When the option to retrieve files in ascending order is selected, user is displayed

with two options of viewing the files.

**i.**  Ascending order of folders first which have files sorted in them,

ii. Ascending order of all files and folders inside the “main” folder.

* 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