LockedMe – Virtual Key for Repositories

BY

SRUTHISREE GUMMADI

Table of contents

1.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 tio digitise their product 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.

- Specification document-Product's Capabilities, appearance, and user interactions
- Number and duration of Sprints required
- Setting 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:
    - Retrieve the file names in 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.

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 developer Details
    - The details of the user interface such as options displaying the user interaction informaiton
    - Features to accept the user 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, collection, and sorting techniques for source code optimisation and increased performance

The requirements should met

- The source code should be pushed to your Github repository.You need to document the steps and write the algorithms in it.
- The submission of your GitHub repository link is mandatory.In order to track your task.You need to share the link of the repository.You can add a section in your document.
- Document the set-by-step process starting from sprint planning to the product release.
- Application should not close, exit or throw an exception if the user specifies an invalid input.
- You need to submit the final specification document which includes
    - Project and developer details
    - Sprints planned and the tasks achieved in them
    - Alogorithms and flowcharts of the application
    - Core concepts used in the project
    - Links to the GitHub repository to verify the project completion
    - Your conclusion on enhancing the application. And defining the USPs (Unique Selling Points)

2.SPRINT PLANNING

This application is planned to complete in 5 sprints.

The sprints are

a.) Developing the flow of application.

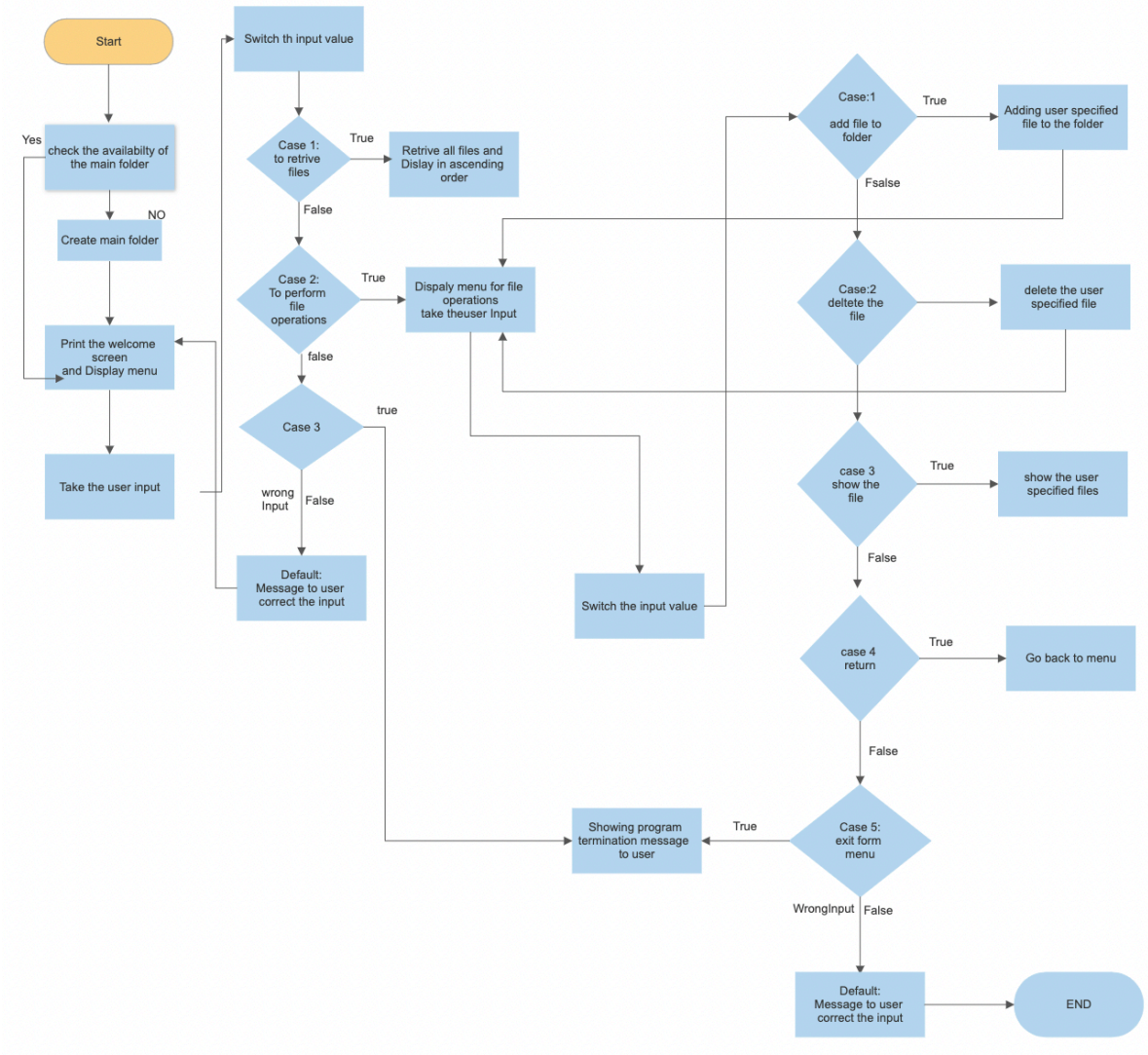
b.)configure the Git Repository to maintain and track the flow and development of the application.

c.) Developing the programs based on the user requirements.

d.) Testing the project with different user aspects.

e.)Documenting about the complete project about its capability and working .

3.FLOW OF APPLICATION



4. CONCEPTS USED IN THE APPLICATION

This project uses file Handling, sorting of files, Recursion, exception handling and flow control and streams IN JAVA programming

5. APPLICATION DETAILS AND USER INTERACTIONS

Demonstrating the product capabilities, appearence and user interactions

The Steps Taken While developing the application

1. Creating a New Project called "Lockers"

   —> Open Eclipse or Spring or any IDE

   —> Go to File—> New—> Project —>JavaProject—> Next

   —> Enter the Project title as "Lockers" then Click on Finish

   —> Select the project and go to File—>New—>class.

   —> write a Main java application program has welcome screen and primary menu options i.e. "**LockedmeMain.java**"

2. Writing a program to perform Primary menu options handling methods i.e

   "PrimaryOptions.java"

3. Writing a program of java class having methods to perform all the file operations such as crating file, deleting file, showallfiles, list them recursively and searching file etc.

   i.e **"FileOPerations.java"**

4. Writing a program to perform user specified operation on main folder having files

   i.e **"SecondaryMenu.java"**

Source code:

LockedmeMain.java

```java
package com;

import java.io.*;
import java.lang.System;

public class LockedmeMain {
    public static void main(String[] args) {
        //printing welcome screen
        //String Locker = String.format("Hello User /n Welcome to LockedMe.com Developed by sruthisree \n");
        String LockerApp ="This application help you to :: \n"
                + "Add or delete files in \"main\" folder \n"
                + "Retrieve all files in the \"main\" folder \n";
        System.out.println("Hello User \n Welcome to LockedMe.com Developed by sruthisree \n");
        System.out.println(LockerApp);
        createFolderIfNotPresent("main");
        // Display options for user
        PrimaryOptions.InputFromWelcomeScreen();
    }
    public static void createFolderIfNotPresent(String folderName) {
            File file = new File(folderName);
            if (!file.exists()) {
                file.mkdirs();
        }

    }

}

```

PrimaryOptions.java

```java
package com;
import java.util.*;

public class PrimaryOptions {

    //handle welcome screen input

    public static void InputFromWelcomeScreen() {
        boolean flow = true;
        Scanner sc = new Scanner(System.in);
        //FileOperations.ShowMenu();
        do {
            try {
                FileOperations.ShowMenu();
                int input = sc.nextInt();

                switch (input) {
                case 1:
                    FileOperations.ShowAllFiles("main");
                    break;
                case 2:
                    FileOperations.FileMenuOperations();
                    SecondaryMenu.SecondaryMenuOptions();
                    break;
                case 3:
                    System.out.println("Program exited successfully.");
                    flow = false;
                    sc.close();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Please select a valid option from above.");
                }
            } catch (Exception e) {
                System.out.println(e.getClass().getName());
                System.out.println("Reload the app and \n then please Enter valid option from 1 to 3 ");
                InputFromWelcomeScreen();
                //break;

            }
        } while (flow == true);

    }
}

```

FileOperation.java

```java
package com;
import java.io.IOException;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.IntStream;
import java.util.Collections;
import java.util.stream.Collectors;
import java.util.Scanner;


public class FileOperations {
    public static void ShowMenu() {
        String menu = "\n --- Select any option number from below and press Enter --\n"
                + "1) Get all files of \"main\" folder\n" + "2) Menu for File operations\n"
                + "3) Exit program\n";
        System.out.println(menu);

    }

    public static void FileMenuOperations() {
        String fileMenu = "\n\n-----Select any option number from below and press Enter -----\n\n"
                + "1) Add a file to \"main\" folder\n" + "2) Delete a file from \"main\" folder\n"
                + "3) Search for a file from \"main\" folder\n" + "4) Show Previous Menu\n" + "5) Exit program\n";

        System.out.println(fileMenu);
    }

    public static void ShowAllFiles(String path) {
        System.out.println("The files with directory structure in ascending order\n");
        listFilesInDirectory(path, 0, new ArrayList<String>());
    }
```

ShowMenu() :: Primary Menu options

FileMenuOerations() :: Menu for file Operations

ShowAllFiles(String path) :: to Display all files in main

```java
    public static List<String> listFilesInDirectory(String path, int Count, List<String> fileList) {
        File dirctr = new File(path);
        File[] files = dirctr.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
                    fileList.add(file.getName());
                    listFilesInDirectory(file.getAbsolutePath(), Count + 1, fileList);
                } else {
                    System.out.println("|** " + file.getName()+"**|");
                    fileList.add(file.getName());
                }
            }
        } else {
            System.out.print(" ".repeat(Count * 2));
            System.out.println("|** Empty Directory **|");
        }
        System.out.println();
        return fileList;
    }

```

listFilesInDirectory():: to sort the files based on their size

```java
public static List<String> FileLocation(String fName, String path) {
    List<String> fileList = new ArrayList<>();
    searchFile(path, fName, fileList);

    if (fileList.isEmpty()) {
        System.out.println("\n\n----No file with given file name \"" + fName + "\" ----\n\n");
    } else {
        System.out.println("\n\n Found the given file at location(s):");

        List<String> files = IntStream.range(0, fileList.size())
                .mapToObj(index -> (index + 1) + ": " + fileList.get(index)).collect(Collectors.toList());

        files.forEach(System.out::println);
    }

    return fileList;
}
```

FileLocation() :: Method to display the Location of the user specified file

```java
public static void createFile(String fAdd, Scanner sc) {
    Path filePath = Paths.get("./main/" + fAdd);
    try {
        Files.createDirectories(filePath.getParent());
        Files.createFile(filePath);
        System.out.println(fAdd + " created successfully");

        System.out.println("Would you like to add some content to the file? (Y/N)");
        String choice = sc.next().toLowerCase();

        sc.nextLine();
        if (choice.equals("y")) {
            System.out.println("\n\nInput content and press enter\n");
            String content = sc.nextLine();
            Files.write(filePath, content.getBytes());
            System.out.println("\nContent written to file " + fAdd);
            System.out.println("Content can be read using Notepad or Notepad++");
        }
        else
        {
            FileMenuOperations();
        }


    } catch (IOException e) {
        System.out.println(" "
                + "File creation failed " + fAdd);
        System.out.println(e.getClass().getName());
    }
}
```

CreateFIle(String,scanner) :: To create user specified file

```java
    public static void FileDelRecursively(String path) {

        File Fdel = new File(path);
        File[] files = Fdel.listFiles();

        if (files != null && files.length > 0) {
            for (File file : files) {

                String fileName = file.getName() + " at " + file.getParent();
                if (file.isDirectory()) {
                    FileDelRecursively(file.getAbsolutePath());
                }

                if (file.delete()) {
                    System.out.println(fileName + " deleted successfully");
                } else {
                    System.out.println("Failed to delete " + fileName);
                }
            }
        }

        String delFile = Fdel.getName() + " at " + Fdel.getParent();
        if (Fdel.delete()) {
            System.out.println(delFile + " deleted successfully");
        } else {
            System.out.println("deletion failed " + delFile);
        }
    }
```

FileDelRecursivley(String) :: To delete user specified file

```java
public static void searchFile(String path, String fName, List<String> fileList) {
    File dir = new File(path);
    File[] files = dir.listFiles();
    List<File> fList = Arrays.asList(files);

    if (files != null && files.length > 0) {
        for (File file : fList) {

            if (file.getName().startsWith(fName)) {
                fileList.add(file.getAbsolutePath());
            }
            if (file.isDirectory()) {
                searchFile(file.getAbsolutePath(), fName, fileList);
            }
        }
    }
```

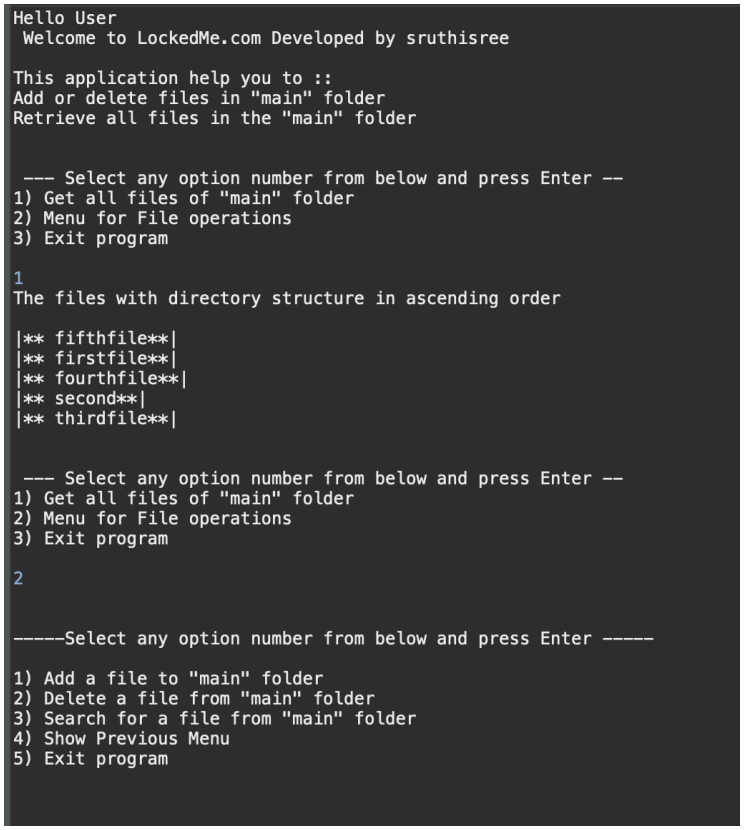
searchFile(String,String,List) :: Method to search file

SecondaryMenu.java

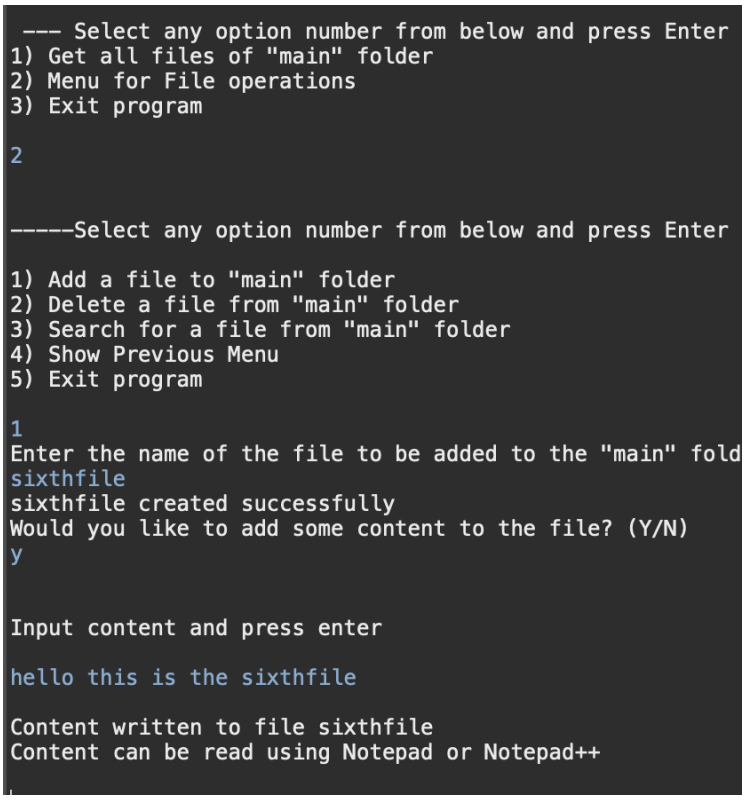
```java
package com;

import java.util.List;
import java.util.Scanner;


public class SecondaryMenu {
    public static void SecondaryMenuOptions(){
        boolean flow = true;
        Scanner sc = new Scanner(System.in);
        do {


            try {

                int input = sc.nextInt();
                switch (input) {

                //case for Adding file
                case 1:

                    System.out.println("Enter the name of the file to be added to the \"main\" folder");
                    String fileToAdd = sc.next();

                    FileOperations.createFile(fileToAdd, sc);

                    break;

                    //case to delete File
                case 2:

                    System.out.println("Enter the name of the file to be deleted from \"main\" folder");
                    String fileToDelete = sc.next();

                    List<String> filesToDelete = FileOperations.FileLocation(fileToDelete, "main");

                    String deletionPrompt = "\nSelect index of which file to delete?"
                            + "\n(Enter 0 if you want to delete all elements)";
                    System.out.println(deletionPrompt);

                    int id = sc.nextInt();

                    if (id != 0) {
                        FileOperations.FileDelRecursively(filesToDelete.get(id - 1));
                    } else {

                        // If id == 0, delete all files displayed for the name
                        for (String path : filesToDelete) {
                            FileOperations.FileDelRecursively(path);
                        }
                    }


                    break;

                    // case to search a File
                case 3:
                    // File/Folder Search
                    System.out.println("Enter the name of the file to be searched from \"main\" folder");
                    String fName = sc.next();
                    FileOperations.FileLocation(fName, "main");
                    PrimaryOptions.InputFromWelcomeScreen();

                    //break;

                    //  Go back to Previous menu
                case 4:
                    return;

                    // case to Exit
                case 5:

                    System.out.println("Program exited successfully.");
                    flow = false;
                    sc.close();
                    System.exit(0);

                default:
                    System.out.println("Please select a valid option from above.");
                }

            } catch (Exception e) {
                System.out.println(e.getClass().getName());
                System.out.println("Reload the app and \n then please Enter valid option from 1 to 5 ");
                break;
            }


        } while (flow == true);


    }

}
```

OUTPUT

WelcomeScreen with Primary and secondary menu options



Adding file



Deleting file and Handling the exception after entering the wrong output

```
2
Enter the name of the file to be deleted from "main" folder
sixthfile


 Found the given file at location(s):
1: /Users/sruthisree/Documents/workspace-spring-tool-suite-4-4.16.1.RELEASE/Lockers/main/sixthfile

Select index of which file to delete?
(Enter 0 if you want to delete all elements)
0
sixthfile at /Users/sruthisree/Documents/workspace-spring-tool-suite-4-4.16.1.RELEASE/Lockers/main deleted successfully
hello
java.util.InputMismatchException
Reload the app and
 then please Enter valid option from 1 to 5

 --- Select any option number from below and press Enter --
1) Get all files of "main" folder
2) Menu for File operations
3) Exit program
```

Searching for file and output verification with handling exceptions

```
-----Select any option number from below and press Enter -----

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
second


 Found the given file at location(s):
1: /Users/sruthisree/Documents/workspace-spring-tool-suite-4-4.16.1.RELEASE/Lockers/main/second

 --- Select any option number from below and press Enter --
1) Get all files of "main" folder
2) Menu for File operations
3) Exit program

2


-----Select any option number from below and press Enter -----

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder

sixth


----No file with given file name "sixth" ----
```

Steps to push the code in git Repositories

Step-1: Create a new GitHub Repository and Goto Directory which has project "Lockers"

$ cd documents/projects/simplilearn/Lockers;

Step-2: Initialise Git in the project folder

$ git init —>enter

Step-3 : Add the files to Git Repository

$ git add -A

Step-4 : Push the files to the folder you initially created using the following command:

$ git push -u origin master

Unique features of the Application

1.This application performs all the Operations on user specified files.

2.This application completely work with user interactions.User can switch to the different inputs.

3.This application is designed to handle all the operations along with the exception handling.

4.The application developed with very less hard coding of data

Conclusion

The application provides all the file operations with completely user interactions.

Future enhancements will amend on concepts of file appendence and file selection based on last modified date and time and content in the file and type of the file.