**FINAL SPECIFICATION DOCUMENT**

**Virtual Key for Your Repositories**

**Table of contents**

1.Project and developer details

2.Sprints planned and the tasks achieved

3.Algorithms and flowcharts of the application

4.Core concepts used in the project

5.Links to the GitHub repository to verify the project completion

6.Conclusion on enhancing the application and defining the USPs (Unique Selling Points)

Name: Aishwarya Sivakumar Manickam

Employee ID: 2527004

1.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.

Problem Statement:

In order to digitize the products of the company Lockers Pvt. Ltd., the developer is asked to develop an application that enables user to add, delete, search file from the application. The goal of the company is to deliver a high-end quality product as early as possible.

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:

Aishwarya Sivakumar is the developer of this application.

Contact: [aishwarya.sivakumar48@gmail.com](mailto:aishwarya.sivakumar48@gmail.com)

2.Sprints planned and the tasks achieved

For developing this application, I have considered 3 sprints each of 3 hrs.

1st 3-hour:

* Deciding the flow of the application
* Defining class names and functions within
* Preparing the eclipse IDE to start code

2nd 3-hour:

* Writing code for the problem statement proposed
* Code is split into several classes and each class output is checked side by side

3rd 3-hour:

* Testing the application that is developed with varies inputs.
* Pushing the code to GitHub
* Creating a word document that specifies the application capabilities and user interactions.

3.Algorithms and flow of the application

3.1 Algorithm and flow

The algorithm and procedures followed to build the project are given as step wise points below. In the same flow , the application is built

Step 1: Firstly, A java project “Phase 1-Assessment” is opened in Eclipse IDE.

Step 2: In the project, a package called lockers\_pvt\_lmt is created.

Step 3: In the project, a public class “Lockers\_pvt\_lmt is defined with it variables.

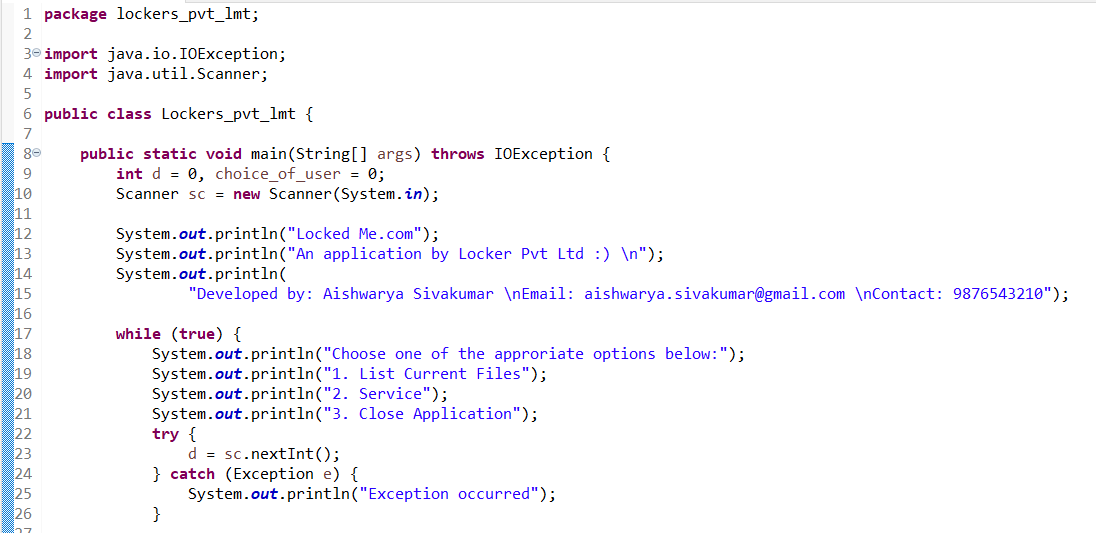
Step 4: After declaring variables, the name of the application and developer details are displayed.

Step 5: While the statements are printed, now user is allowed to choose some options that are listed in the console.

Step 6: They are 1. List current files 2. Service 3. Close application

Step 7: If the user entered any other option other than 3, try catch block is used to catch the error and execute further.

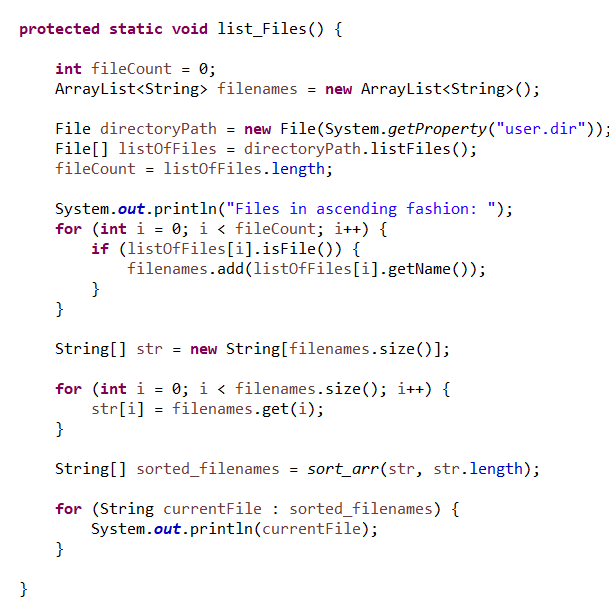
Step 8: While options being correct, respective functions will execute further.



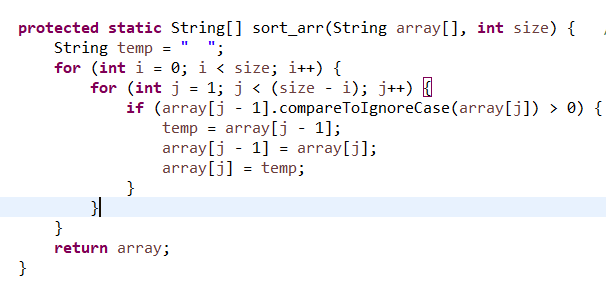
Step 9: The functions to be performed are given in switch case.

Step 10: Case 1: If the user entered option 1 i.e., List current files, respective function will be called and executed. It returns the list of files in the folder.

Step 11: The function name is list\_files where an array List is created to fetch the list of files from the mentioned path.

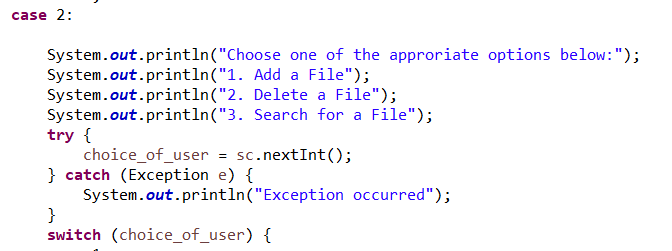


Step 12: In order to list the files in ascending order, Bubble sort technique is initiated to obtain the list of files.

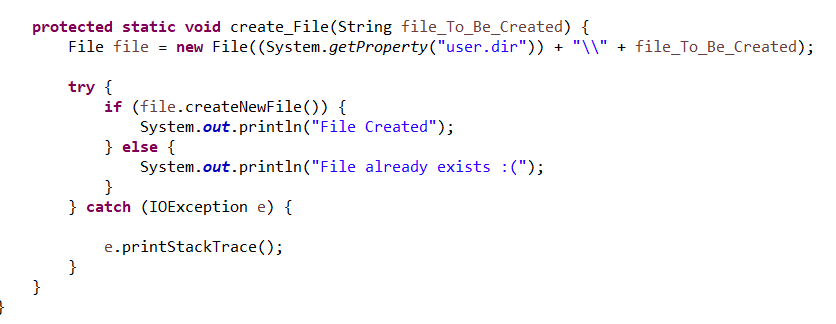


Step 13: Case 2: If the user entered option 2 i.e., Service, respective function will be called and executed. It further displays 3 functions namely 1. Add a File, 2. Delete a File 3. Search for a file.

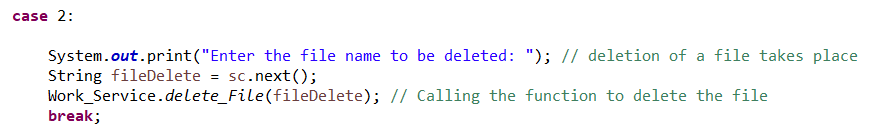
Step 14: These three sub-options are further given a switch case to go for an exact function that a user wants.



Step 15: In the service section, if the user entered option 1 i.e, add a file, Create\_File function is called where file name is entered by the user and it is created.

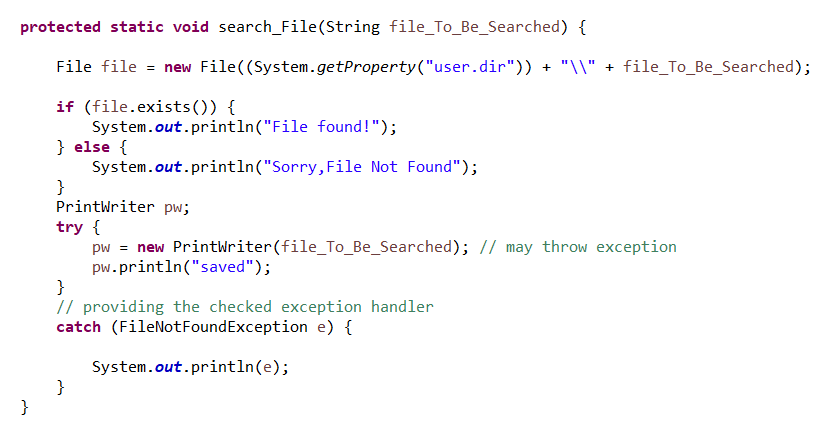


Step 16: In the service section, if the user entered option 2 i.e., Delete file, delete\_file is called where user defined function file. Delete from java.io. File; is used to delete the file entered by the user.



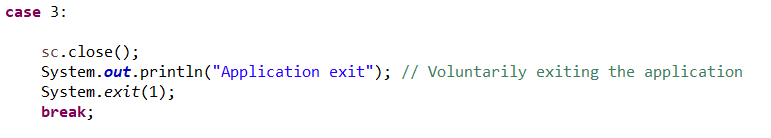


Step 17: In the service section, if the user entered option 3 i.e., Search for a file, Search\_file function is called where the file is checked for its existence the respective directory. If found it displays” file found” else it displays “file not found”.



Step 18: After performing any of these options it returns back to main options that is list current files, service and close application.

Step 19: Case 3: If the user entered option 3 i.e., Close application, it exists the application.



3.2 Code

**package** lockers\_pvt\_lmt;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** Lockers\_pvt\_lmt {

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

**int** d = 0, choice\_of\_user = 0;

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

System.***out***.println("Locked Me.com");

System.***out***.println("An application by Locker Pvt Ltd :) \n");

System.***out***.println(

"Developed by: Aishwarya Sivakumar \nEmail: aishwarya.sivakumar@gmail.com \nContact: 9876543210");

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

System.***out***.println("Choose one of the approriate options below:");

System.***out***.println("1. List Current Files");

System.***out***.println("2. Service");

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

**try** {

d = sc.nextInt();

} **catch** (Exception e) {

System.***out***.println("Exception occurred");

}

**switch** (d) {

**case** 1:

Work\_Service.*list\_Files*();

**break**;

**case** 2:

System.***out***.println("Choose one of the approriate options below:");

System.***out***.println("1. Add a File");

System.***out***.println("2. Delete a File");

System.***out***.println("3. Search for a File");

**try** {

choice\_of\_user = sc.nextInt();

} **catch** (Exception e) {

System.***out***.println("Exception occurred");

}

**switch** (choice\_of\_user) {

**case** 1:

System.***out***.println("Enter the file name to be created: "); // Creation of a file takes place

String fileCreate = sc.next();

Work\_Service.*create\_File*(fileCreate); // Calling the function to create the file

**break**;

**case** 2:

System.***out***.print("Enter the file name to be deleted: "); // deletion of a file takes place

String fileDelete = sc.next();

Work\_Service.*delete\_File*(fileDelete); // Calling the function to delete the file

**break**;

**case** 3:

System.***out***.println("Enter the file name to be searched: "); // Search for a file takes place

String fileSearch = sc.next();

Work\_Service.*search\_File*(fileSearch); // Calling the function to search the file

**break**;

**default**:

System.***out***.println("Invalid Input,Repeat the process"); // In the case of unprecedented input // execute this

**break**;

}

**break**;

**case** 3:

sc.close();

System.***out***.println("Application exit"); // Voluntarily exiting the application

System.*exit*(1);

**break**;

**default**:

System.***out***.println("Invalid Input, Select within the range of 1-3"); // In the case of unprecedented // input execute this

**break**;

}

}

}

}

**package** lockers\_pvt\_lmt;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**public** **class** Work\_Service {

**protected** **static** String[] sort\_arr(String array[], **int** size) { // sorting function

String temp = " ";

**for** (**int** i = 0; i < size; i++) {

**for** (**int** j = 1; j < (size - i); j++) {

**if** (array[j - 1].compareToIgnoreCase(array[j]) > 0) {

temp = array[j - 1];

array[j - 1] = array[j];

array[j] = temp;

}

}

}

**return** array;

}

**protected** **static** **void** list\_Files() {

**int** fileCount = 0;

ArrayList<String> filenames = **new** ArrayList<String>();

File directoryPath = **new** File(System.*getProperty*("user.dir"));

File[] listOfFiles = directoryPath.listFiles();

fileCount = listOfFiles.length;

System.***out***.println("Files in ascending fashion: ");

**for** (**int** i = 0; i < fileCount; i++) {

**if** (listOfFiles[i].isFile()) {

filenames.add(listOfFiles[i].getName());

}

}

String[] str = **new** String[filenames.size()];

**for** (**int** i = 0; i < filenames.size(); i++) {

str[i] = filenames.get(i);

}

String[] sorted\_filenames = *sort\_arr*(str, str.length);

**for** (String currentFile : sorted\_filenames) {

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

}

}

**protected** **static** **void** delete\_File(String file\_To\_Be\_Deleted) {

File file = **new** File((System.*getProperty*("user.dir")) + "\\" + file\_To\_Be\_Deleted);

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

**if** (file.delete()) {

System.***out***.println("File deleted successfully!");

}

} **else** {

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

}

}

**protected** **static** **void** search\_File(String file\_To\_Be\_Searched) {

File file = **new** File((System.*getProperty*("user.dir")) + "\\" + file\_To\_Be\_Searched);

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

System.***out***.println("File found!");

} **else** {

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

}

PrintWriter pw;

**try** {

pw = **new** PrintWriter(file\_To\_Be\_Searched); // may throw exception

pw.println("saved");

}

// providing the checked exception handler

**catch** (FileNotFoundException e) {

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

}

}

**protected** **static** **void** create\_File(String file\_To\_Be\_Created) {

File file = **new** File((System.*getProperty*("user.dir")) + "\\" + file\_To\_Be\_Created);

**try** {

**if** (file.createNewFile()) {

System.***out***.println("File Created");

} **else** {

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

}

} **catch** (IOException e) {

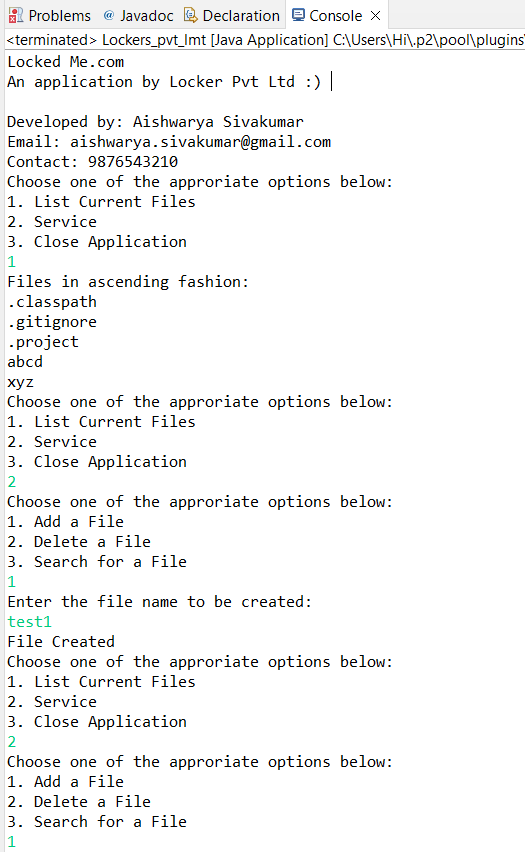
e.printStackTrace();

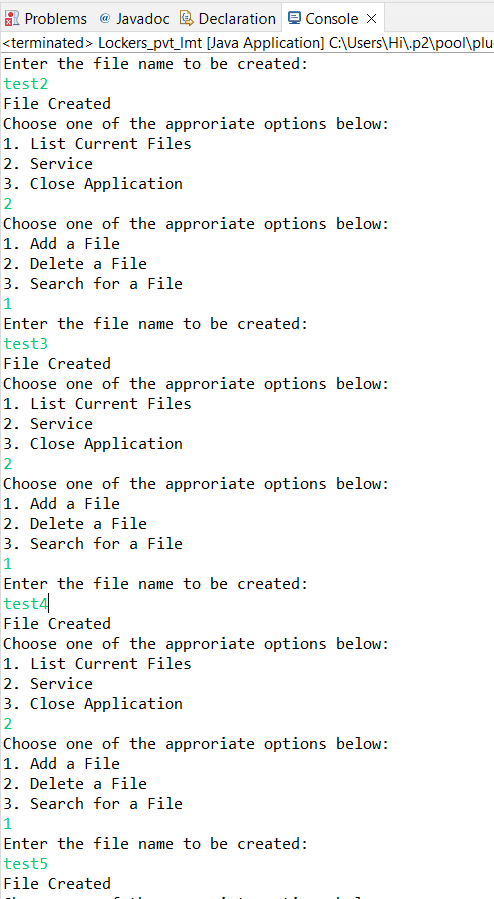
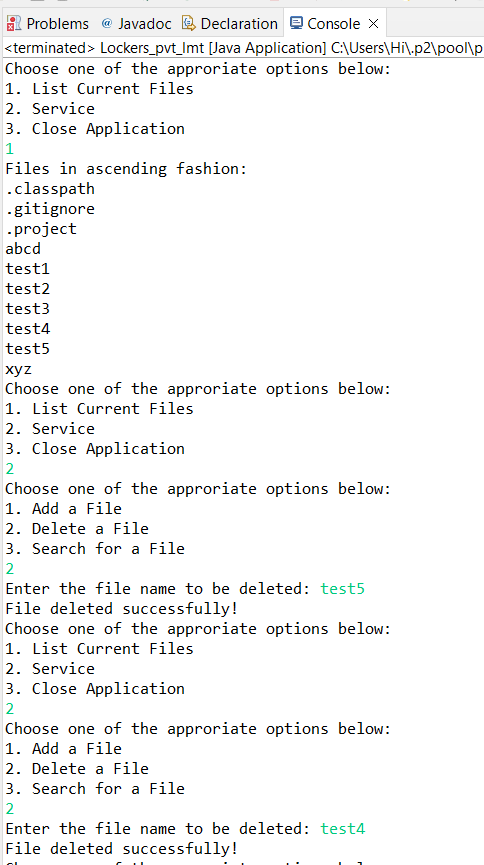
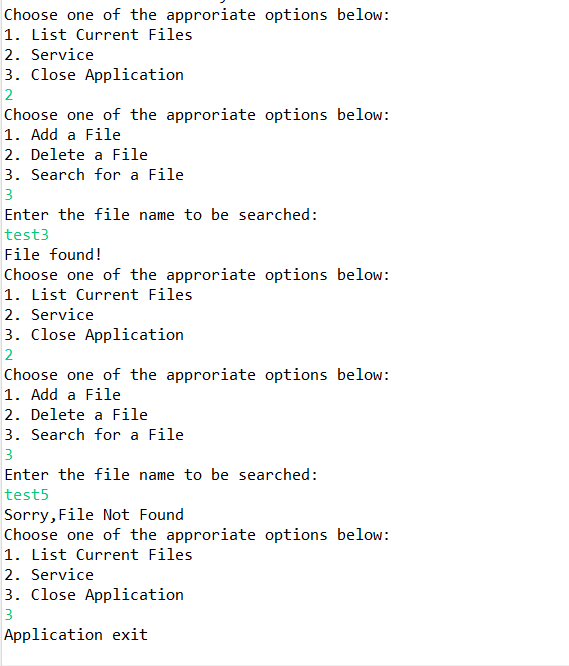
}

}

}

3.3 Output Screenshots



4.Core concepts used in the project

The core concepts used to develop the project are listed below

* Core java
* Collections
* File Handling
* Sorting
* Flow Control
* Recursion
* Exceptional Handling

5.Links to the GitHub repository to verify the project completion

The project phase-1 assessment is pushed to git repository.

The link is provided below

<https://github.com/AishwaryaSivakumar48/MyJavafsd/tree/main/Phase1-Assesssment/src/lockers_pvt_lmt>

6.Conclusion

* User can be enabled to write some context in file after file creation
* User can be enabled to append to the file.
* User can be enabled to know the type of the file like .txt/.pdf etc..