**PHASE 1 FINAL PROJECT: Virtual Key for Your Repositories**

**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. 

**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 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

**You must use the following:**

* Eclipse/IntelliJ: An IDE to code for the application
* Java: A programming language to develop the prototype
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Search and Sort techniques: Data structures used for the project
* Specification document: Any open-source document or Google Docs

**Following requirements should be 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 step-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
  + Algorithms 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)

**Creating a new project in Eclipse:**

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in project name as **FinalProject-Phase1** and click on “Finish.”
* Select your project and go to File -> New -> Package.
* Enter **com.simplilearn.test** as Package name.
* Select your project and go to File -> New -> Class.
* Enter **LockedMe** as class name, check the checkbox “public static void main(String[] args)”., and click on “Finish.”

**package** com.simplilearn.test;

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**public** **class** LockedMe {

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

*welcomeScreen*();

**try** {

*menuDriven*();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

}

**public** **static** **void** welcomeScreen() {

System.***out***.println("Application Name : LOCKED ME \n");

System.***out***.println("Developer Details: Developed by Posi Papa Rohini Eli\n");

}

**public** **static** **void** menuDriven() **throws** FileNotFoundException {

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

BusinessLevelOperations obj = **new** BusinessLevelOperations();

**int** option;

**do** {

System.***out***.println("Enter your choice which you want to select: \n");

System.***out***.println("\t1. Retrieve current filenames in ascending order \n");

System.***out***.println("\t2. Business-level operation menu \n");

System.***out***.println("\t3. Exit from the application \n");

option=sc.nextInt();

**switch**(option) {

**case** 1:

obj.showAllFiles();

**break**;

**case** 2:

**int** ch;

**do** {

System.***out***.println("Enter your choice for Business level operation");

System.***out***.println("\t1. Add a file and its content to a directory");

System.***out***.println("\t2. Delete a file from a directory");

System.***out***.println("\t3. Searching a file and showing its content");

System.***out***.println("\t4. Exit from BLO menu");

ch=sc.nextInt();

**switch**(ch) {

**case** 1:

obj.addFile();

**break**;

**case** 2:

obj.deleteFile();

**break**;

**case** 3:

obj.searchFile();

**break**;

**case** 4:

System.***out***.println("Exited from the Business Level operation...");

System.***out***.println("---------------------------------\n");

**break**;

**default**:

System.***out***.println("Invalid Choice");

**break**;

}

}**while**(ch!=4);

**break**;

**case** 3:

System.***out***.println("Exiting from the application...");

**break**;

**default**:

System.***out***.println("Invalid choice");

}

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

}

}

**package** com.simplilearn.test;

**import** java.awt.Desktop;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** BusinessLevelOperations **implements** FileInterface {

List<String> retrieve= **new** ArrayList<String>();

File[] files = **new** File("D:\\Myprojects\\Workspace\\FinalProject-Phase1").listFiles();

**public** **void** showAllFiles() {

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

**if**(file.isFile()) {

retrieve.add(file.getName());

}

}

retrieve.forEach(System.***out***::println);

}

**public** **void** addFile() {

System.***out***.println("Enter the file which you want to add:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**try** {

**if**(F.createNewFile()) {

System.***out***.println(filename+" file is added to the directory");

Desktop.*getDesktop*().edit(F);

}

**else** {

System.***out***.println("This file is already there");

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** deleteFile() {

System.***out***.println("Enter the file which you want to delete:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**if**(F.delete())

System.***out***.println(filename+" got Deleted");

**else**

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

}

**public** **void** searchFile() {

//

**try** {

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

File directory = **new** File("D:\\\\Myprojects\\\\Workspace\\\\FinalProject-Phase1");

System.***out***.println("Enter the file name which you want to search:");

String fileName=scan.nextLine();

File[] files=directory.listFiles();

**int** flag=0;

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

String name = file.getName();

**if** (name.equals(fileName)) {

File f= **new** File(fileName);

Scanner sc1 = **new** Scanner(f);

**while**(sc1.hasNextLine()) {

System.***out***.println(sc1.nextLine());

}

flag=1;

}

}

**if**(flag==0) {

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

}

}**catch**(FileNotFoundException ex) {

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

}

}

}

**package** com.simplilearn.test;

**public** **interface** FileInterface {

**public** **void** showAllFiles();

**public** **void** addFile();

**public** **void** deleteFile();

**abstract** **void** searchFile();

}

**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

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

*welcomeScreen*();

**try** {

*menuDriven*();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

}

**public** **static** **void** welcomeScreen() {

System.***out***.println("Application Name : LOCKED ME \n");

System.***out***.println("Developer Details: Developed by Posi Papa Rohini Eli\n");

}

**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 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.

**public** **void** showAllFiles() {

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

**if**(file.isFile()) {

retrieve.add(file.getName());

}

}

retrieve.forEach(System.***out***::println);

}

**The second option should return the details of the user interface such as options displaying the following:**

**BusinessLevelOperation:**

**package** com.simplilearn.test;

**import** java.awt.Desktop;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** BusinessLevelOperations **implements** FileInterface {

List<String> retrieve= **new** ArrayList<String>();

File[] files = **new** File("D:\\Myprojects\\Workspace\\FinalProject-Phase1").listFiles();

**public** **void** showAllFiles() {

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

**if**(file.isFile()) {

retrieve.add(file.getName());

}

}

retrieve.forEach(System.***out***::println);

}

**public** **void** addFile() {

System.***out***.println("Enter the file which you want to add:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**try** {

**if**(F.createNewFile()) {

System.***out***.println(filename+" file is added to the directory");

Desktop.*getDesktop*().edit(F);

}

**else** {

System.***out***.println("This file is already there");

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **void** deleteFile() {

System.***out***.println("Enter the file which you want to delete:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**if**(F.delete())

System.***out***.println(filename+" got Deleted");

**else**

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

}

**public** **void** searchFile() {

//

**try** {

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

File directory = **new** File("D:\\\\Myprojects\\\\Workspace\\\\FinalProject-Phase1");

System.***out***.println("Enter the file name which you want to search:");

String fileName=scan.nextLine();

File[] files=directory.listFiles();

**int** flag=0;

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

String name = file.getName();

**if** (name.equals(fileName)) {

File f= **new** File(fileName);

Scanner sc1 = **new** Scanner(f);

**while**(sc1.hasNextLine()) {

System.***out***.println(sc1.nextLine());

}

flag=1;

}

}

**if**(flag==0) {

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

}

}**catch**(FileNotFoundException ex) {

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

}

}

}

**package** com.simplilearn.test;

**public** **interface** FileInterface {

**public** **void** showAllFiles();

**public** **void** addFile();

**public** **void** deleteFile();

**abstract** **void** searchFile();

}

Add a file to the existing directory list

Add a file to the existing directory list

You can ignore the case sensitivity of the file names

**public** **void** addFile() {

System.***out***.println("Enter the file which you want to add:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**try** {

**if**(F.createNewFile()) {

System.***out***.println(filename+" file is added to the directory");

Desktop.*getDesktop*().edit(F);

}

**else** {

System.***out***.println("This file is already there");

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

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)

**public** **void** deleteFile() {

System.***out***.println("Enter the file which you want to delete:");

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

String filename=scan.nextLine();

File F= **new** File(filename);

**if**(F.delete())

System.***out***.println(filename+" got Deleted");

**else**

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

**public** **void** searchFile() {

//

**try** {

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

File directory = **new** File("D:\\\\Myprojects\\\\Workspace\\\\FinalProject-Phase1");

System.***out***.println("Enter the file name which you want to search:");

String fileName=scan.nextLine();

File[] files=directory.listFiles();

**int** flag=0;

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

String name = file.getName();

**if** (name.equals(fileName)) {

File f= **new** File(fileName);

Scanner sc1 = **new** Scanner(f);

**while**(sc1.hasNextLine()) {

System.***out***.println(sc1.nextLine());

}

flag=1;

}

}

**if**(flag==0) {

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

}

}**catch**(FileNotFoundException ex) {

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

}

}

Option to navigate back to the main context

Option to navigate back to the main context

* **case** 4:
* System.***out***.println("Exited from the Business Level operation...");
* System.***out***.println("---------------------------------\n");
* **break**;

**There should be a third option to close the application**

System.***out***.println("Exiting from the application...");

**Pushing the code to your GitHub repositories:**

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your 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 “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**