**Simple Learn – Phase 1 Project**

**Virtual Key for Your Repositories**

**Specification Document**

**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.

**GitHub Repository:** [*https://github.com/Prem-Ravi/LockedMe.com.git*](https://github.com/Prem-Ravi/LockedMe.com.git)

**SPRINTS PLANNING:**

**SPRINT 1:**

* Setup the workspace in eclipse IDE.
* Creating GitHub Repository to keep the project in track.
* Flow of the application is designed.
* Java Core concepts to be implemented are planned.

**SPRINT 2:**

* Java Program to implement all the concepts as per the Flow plan.
* Execution of the program as per the requirements.
* Testing the program with different inputs
* Final corrections are done as the application won’t close, exit, or throw an exception if the user specifies an invalid input.

**SPRINT 3:**

* Pushing the code to the GitHub Repository.
* Creating documents like specification document, source code and output screenshots.
* Final check and completion of the project.

**FLOW OF THE APPLICATION:**

Diagram

Description automatically generated

**CORE CONCEPTS USED IN THE PROJECT:**

* Collections Framework
* File Handling
* Sorting
* Flow Control
* Recursion
* Exception Handling

**PRODUCT CAPABILITIES, APPEARANCE, AND USER INTERACTIONS:**

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1. Creating the project in Eclipse
2. Writing a program in Java for the entry point to list the features of the application

(**Main.java**)

1. Writing a program in Java to work on the features listed in main.java

(**BusinessOperations.java**)

1. Pushing the code to GitHub repository

**CREATING A NEW PROJECT IN ECLIPSE:**

* + Open Eclipse
  + Go to File -> New -> Project -> Java Project -> Next.
  + Type in any project name and click on “Finish.”
  + Select your project and go to File -> New -> Class.
  + Enter **Main** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

**WRITING A PROGRAM IN JAVA FOR THE ENTRY POINT OF THE APPLICATION** (**LockedMeMain.java**)

**package** VirtualKeyforYourRepositories;

**import** java.io.IOException;

**import** java.util.Scanner;

**public** **class** Main {

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

**int** ch=0, choice=0;

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

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

System.***out***.println("\t\t LockedMe.com ");

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

System.***out***.println("\tDeveloper\t: Prem Ravi \n \tCompany \t: Lockers Pvt. Ltd. \n \tProduct \t: SL Phase 1 Project \n");

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

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

{

System.***out***.println("Features: \n");

System.***out***.println("1. Retrive Files");

System.***out***.println("2. Business Operations");

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

System.***out***.println("Choose one of the above feature:");

**try**

{

ch = sc.nextInt();

}

**catch**(Exception e)

{

System.***err***.println("Sorry! Please type a number between 1 and 3");

**break**;

}

**switch**(ch)

{

**case** 1: //List function feature to list all files in ascending order.

BusinessOperations.*listFiles*();

**break**;

**case** 2:

**boolean** flag=**true**;

**while**(flag) {

System.***out***.println("Please choose one of the following options :");

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

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

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

System.***out***.println("4. Navigate to previous Menu options");

**try**{

choice = sc.nextInt();

}

**catch**(Exception e)

{

System.***err***.println("Sorry! Please type a number between 1 and 3");

**break**;

}

**switch**(choice)

{

**case** 1:

//Creation of a file takes place

System.***out***.println("Input the name of a file to be created: ");

String fileCreate = sc.next();

// Calling the function to create the file

BusinessOperations.*createFile*(fileCreate);

**break**;

**case** 2:

//deletion of a file takes place

System.***out***.print("Input the name of a file to be deleted: ");

String fileDelete = sc.next();

// Calling the function to delete the file

BusinessOperations.*deleteFile*(fileDelete);

**break**;

**case** 3:

//Search for a file takes place

System.***out***.println("Input the name of a file to be searched: ");

String fileSearch = sc.next();

// Calling the function to search the file

BusinessOperations.*searchFile*(fileSearch);

**break**;

**case** 4:

flag = **false**;

**break**;

**default**:

//In the case of unprecedented input execute this

System.***err***.println("\n Opps! Invalid Input,Re-do the process\n");

**break**;

}

}

**break**;

**case** 3:

//Voluntarily exiting the application

sc.close();

System.***out***.println("\nIt was nice having you here! See you again. Good bye...");

System.*exit*(1);

**break**;

**default**:

//In the case of unprecedented input execute this

System.***err***.println("\n\n Opps! Invalid Input, Select within the range of 1-3\n");

**break**;

}

}

}

}

**WRITING A PROGRAM IN JAVA TO WORK ON THE FEATURES LISTED IN main.java**

(**BusinessOperations.java**)

**package** VirtualKeyforYourRepositories;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** BusinessOperations {

**public** **static** String[] sort\_sub(String array[], **int** size) {

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;

}

// File listing function

**public** **static** **void** listFiles() {

**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 order: ");

**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\_sub*(str, str.length);

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

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

}

}

// File delete function

**public** **static** **void** deleteFile(String fileToBeDeleted) {

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

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

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

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

}

} **else** {

System.***out***.println("Sorry, File wasn't deleted (File Not Found)");

}

}

// File search function

**public** **static** **void** searchFile(String fileToBeSearched) {

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

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

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

} **else** {

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

}

PrintWriter pw;

**try** {

pw = **new** PrintWriter(fileToBeSearched); // may throw exception

pw.println("saved");

}

// providing the checked exception handler

**catch** (FileNotFoundException e) {

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

}

}

// File creation function

**public** **static** **void** createFile(String fileToBeCreated) {

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

**try** {

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

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

}

**else** {

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

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

Welcome Screen of LockedMe.com

**Table

Description automatically generated**

Retriving files in ascending order.

****

Adding New File

**Text

Description automatically generated**

Deletion of a file

**Text

Description automatically generated**

Search of a file

**Text

Description automatically generated**

Navigate to previous menu

**Graphical user interface, text, application

Description automatically generated**

Close the Application.

**Text

Description automatically generated**

Files present in disk.

**Table

Description automatically generated**

**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 done>**

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

**git push -u origin main**

## **UNIQUE SELLING POINTS OF THE APPLICATION:**

* The application is made to continue functioning and accepting user input even in the face of errors. The proper option must be chosen in order to terminate the application.
* The application can take any file/folder name as input. The user can specify a relative path and the application will create the necessary folder structure even if the user wants to create nested folder structures.
* Additionally, the user is given the choice to add content to the newly generated file if they so choose.
* 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 user 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 is performed.
* When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.
* Ascending order of folders first which have files sorted in them,
* 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.