**PHASE1 PROJECT**

**VIRTUAL KEY FOR YOUR REPOSITORIES**

* This document contains sections for:
* Sprint planning and Task completion
* Core concepts used in project
* Flow of the Application.
* Demonstrating the product capabilities, appearance, and user interactions.
* Unique Selling Points of the Application
* Conclusions

*The code for this project is hosted* [PHASE1/Locker-Virtualkey/src/lockercompany at main · Nandy-M/PHASE1 (github.com)](https://github.com/Nandy-M/PHASE1/tree/main/Locker-Virtualkey/src/lockercompany)*.*

*The project is developed by Nandhini.M*

**SPRINT PLANNING AND TASK COMPLETION**

The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

* Creating the flow of the application
* Initializing git repository to track changes as development progresses.
* Writing the Java program to fulfill the requirements of the project.
* Testing the Java program with different kinds of User input
* Pushing code to GitHub.
* Creating this specification document highlighting application capabilities, appearance, and user interactions.

**CORE CONCEPTS USED IN PROJECT**

* Class and Object
* Control Statement
* File Handling
* Exception Handling
* Collections (ArrayList)

**FLOW OF THE APPLICATION**

Start

Print Welcome screen and

Application Function

Display Initial Main menu

Take the choice from User

True

Switch Input value Case1 Retreive all the files and display it in ascending order

Case2 Display choice for Business Level Operations

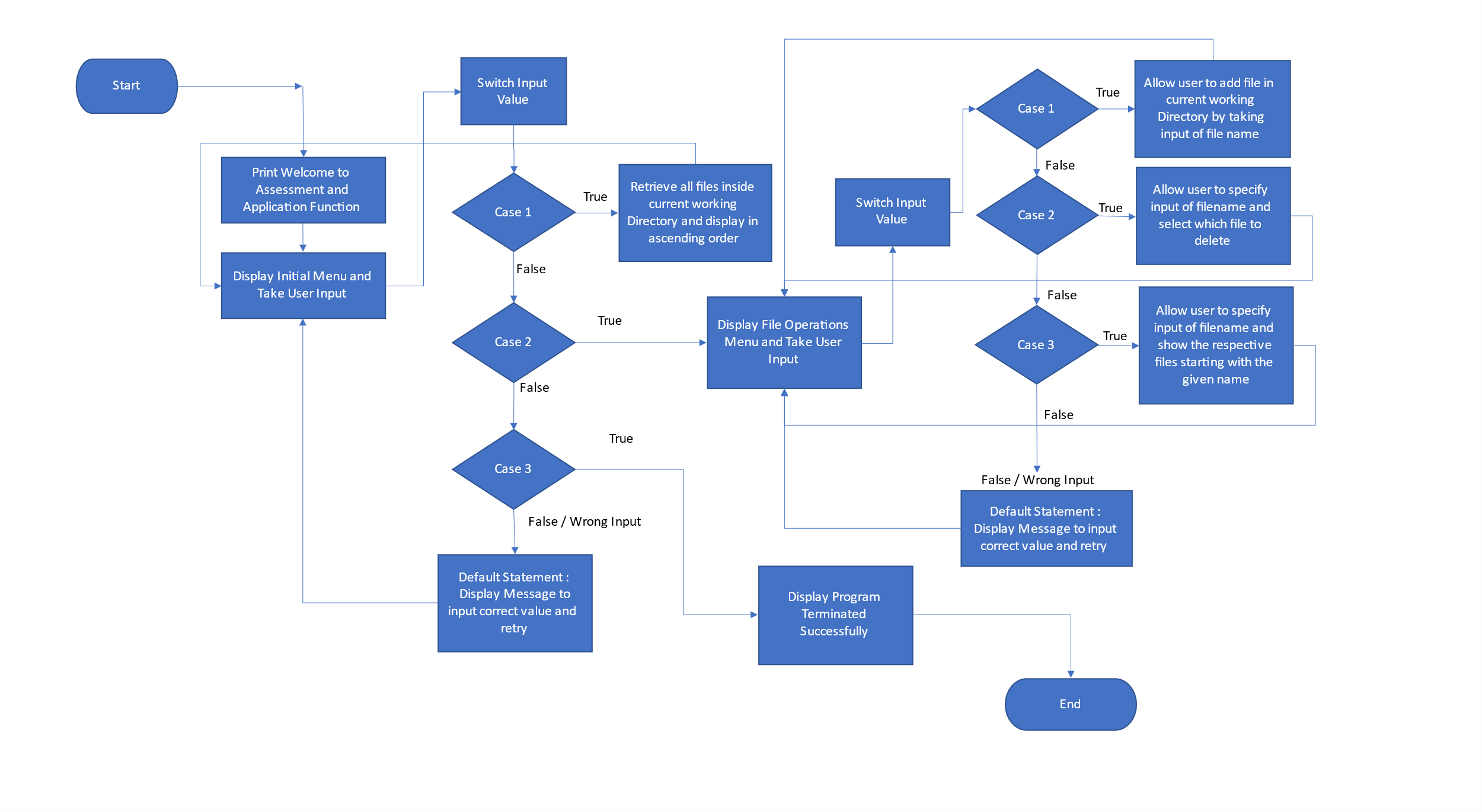
1.add the file 2. Delete the file 3.Search for file

4.Navigate to main menu 5. Terminate Program

Case3 Close the application

False

Default will executed and says invalid input.



**SPECIFICATION DOCUMENT**

Below are the sub-sections set up to highlight project aesthetics and user interactions in order to demonstrate the capabilities of the product:

* **PRODUCT’S CAPABILITIES**
* **APPEARANCE**
* **USER INTERACTIONS**
* [Creating the New project in Eclipse](#Step_1)
* [Writing a program in Java for the entry point of the application (MainDrive.java)](#Step_2)
* [Writing a program in Java to display the Current file List Option available for the user (DisplayFiles.java)](#Step_3)
* [Writing a program in Java to handle Business Operation like Add, delete, search options selected by user (BusinessOperations.java)](#Step_4)
* [Pushing the code to GitHub repository](#Step_6)

**STEP 1** [Creating the New project in Eclipse](#Step_1)

* 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 **MainDrive** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

**STEP 2** [Writing a program in Java for the entry point of the application (MainDrive.java)](#Step_2)

* Select your project and go to File -> New -> Class.
* Enter MainDrive in class name and click on “Finish.”

MainDrive consists of-:

* Details of the Developer
* Displaying Welcome screen to the user

package lockercompany;

import java.io.IOException;

import java.util.Scanner;

public class MainDrive {

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

int menu\_choise = 0, operations\_choise = 0;

Scanner sc = new Scanner(System.in);

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

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

System.out.println("\t By Lockers Pvt Ltd. \n");

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

System.out.println(" Developed by \t: Nandhini M \n Company\t: Simplilearn \n");

* Here, Display the Choise for Main Menu Options

while (true) {

System.out.println("\*\*\*\*\*\*\*\*\*\*Prototype of the Application LockedMe.com\*\*\*\*\*\*\*\*\*\* \n");

System.out.println("---->choose one of the Menu options---->");

System.out.println("1. Display Current Files");

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

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

try {

menu\_choise = sc.nextInt();

} catch (Exception e) {

System.out.println("Exception raised:Null Exception occurred");

}

* Display the option for Business level operations

switch (menu\_choise) {

case 1:

DisplayFiles.listFiles();

break;

case 2:

Boolean temp = true;

while (temp) {

System.out.println("---->choose one of the BusinessLevel options---->");

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

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

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

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

System.out.println("Option 5. Terminate Program");

try {

operations\_choise = sc.nextInt();

} catch (Exception e) {

System.out.println("Exception raised:Null Exception occurred");

}

switch (operations\_choise) {

case 1:

System.out.println("Put 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("Put 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("Put 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:

temp = false;

break;

case 5:

System.out.println("/tSuccessfully! Program Terminated ");

System.exit(0);

default:

System.out.println("\n Opps! Invalid Input,Try again the process\n");

break;

}

}

break;

* Then Closing application will execute for if we given the respective choise

case 3:

// Voluntarily exiting the application

sc.close();

System.out.println("\n It was nice having you here! Happy to See you again. Good bye...");

System.exit(1);

break;

default:

// In the case of unprecedented input execute this

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

break;

}

}

}

}

* STEP 3 [Writing a program in Java to display the Current file Option available for the user (DisplayFiles.java)](#Step_3)
* Select your project and go to File -> New -> Class.
* Enter DisplayFiles in class name and click on “Finish.”
* Displaying all the files in ascending order

package lockercompany;

import java.io.File;

import java.util.ArrayList;

public class DisplayFiles {

//Here Using bubble sort string array will be sorted

protected static String[] Sorting(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

protected static void listFiles() {

int CountFile = 0;

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

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

File[] listOfFiles = CurrentdirectoryPath.listFiles();

CountFile = listOfFiles.length;

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

for (int i = 0; i < CountFile; 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 = Sorting(str, str.length);

for(String currentFile: sorted\_filenames) {

System.out.println(currentFile);

}

}

}

* STEP 4 [Writing a program in Java to handle Business Operation Display options selected by user (BusinessOperations.java)](#Step_4)
* Select your project and go to File -> New -> Class.
* Enter BusinessOperations in class name and click on “Finish.”
* In this class,There is a method for file creating,file deleting,file searching

package lockercompany;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.PrintWriter;

public class BusinessOperations {

//File creation function

protected 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();

}

}

//File delete function

protected static void deleteFile(String fileToBeDeleted) {

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

if(file.exists()) {

if ( file.delete() ) {

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

}

} else {

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

}

}

//File search function

protected 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("Filesaved"); }

catch (FileNotFoundException e) {

System.out.println(e);

}

}

}

* STEP 5 [Pushing the code to GitHub repository](#Step_6)
* 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 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.