

Implement OOPS using JAVA with Data Structures and Beyond

Project 1

Virtual Key for Your Repository

Project Objectives:

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);

Link to GitHub

https://github.com/Drazen-BBG/SimpliLearn_OOPS_Project

Sprint Planning

This project was divided into 5 sprints, and the goal is to have a working console app the whole time.

Sprint 1:

Creating an IntelliJ IDEA project, setting up a git repository and pushing it to GitHub. The goal is to have a working app the whole time and a very nice presentation in the console. So the 1st sprint was all about creating `welcomeMessage()` and `goodbyeMessage()` and presenting it nicely in the console using two methods for nice and even presentation in the console named `lineBuilderCenterAligned` and `lineBuilderLeftAligned()`. Also few variables to store the data are needed and those are `appName`, `consoleBoxWidth` for example.

The second part of the first sprint was the creation of the `mainMenu()` and the ability to set the `absoluteAppDirectoryPath` and the `currentDirectoryPath` with the all needed methods related to them. The goal was to be able to set and get the path to the folder in which the app will be exploring, creating, sorting, searching and deleting the files. For this purpose many methods and variables need to be created like: `setAbsoluteAppDirectoryPath()`, `getAbsoluteAppDirectoryPath()`, `setCurrentDirectoryPath()`, `getCurrentDirectoryPath()` those are the simple setters and getters. Later we will implement better solutions.

The third part was all about the user input and its validation. For this purpose we need `userInput()` which will get `getUserInput()` in form of a `String` and then we need to validate if it is an integer and is it in range so two more methods are needed and those are `isInteger()` and `isInRange()`. Now the `mainMenu` was able to perform few tasks and it was time to plan the 2nd sprint.

Sprite 2:

In the second sprint we want to add some functionality to our `mainMenu()` so we will start with simple stuff first. We will create the ability to show of by creating a `developerInfo()` method and its variables to be able to present the developer to the audience. For this purpose, few more variables are need to store the info `developerName`, `developerLinkedin` and `gitHubAppLink`.

In the second part of the 2nd sprint the goal is to be able to `startApp()` and `exitApp()` and of course to delay the code execution for better presentation so we need to create `waitSeconds()` method for this purpose.

Sprint 3:

In the third spring we will focus on the `fileExplorer()` and its `fileDetailsMessage()` methods. For its implementation we will need to develop few methods to support the cause and few of those are going to be `getFileListFromFolder()` to be able to list files from a specific folder path and `fileDetails()` to be able to present details for a specific file or folder and for its better presentation we will create `fileDetailsMessage()` to separate message from functionality as we did before.

In the second part of the 3rd sprint we will develop `goIntoDirectory()` method to be able to enter directories and `goToParentDirectory()` method to be able to go to a parent directory. By now we will have nice `mainMenu()` and `fileExplorer` where we can go through folders and see details for selected files. The only remaining thing to do in this sprint is to implement the ability of creating files in a specific folder using `fileExplorer()` and its sub methods `crateFile()` and `crateFileMessage()` for separation of the code. Since we can `crateFile()` now we will add the option to delete the file to using `deleteFileProtocol()` method which uses `deleteFileProtocolMessage` for its presentation and for its execution we will create `deleteFileExecution()` method.

Sprite 4:

Our app needs to keep developing and in the 4th sprite we will add the remaining needed functionality and that is sorting and searching. First we will refactor the code and do a nice comment markup of the whole code before we go for implementation of the remaining functionality.

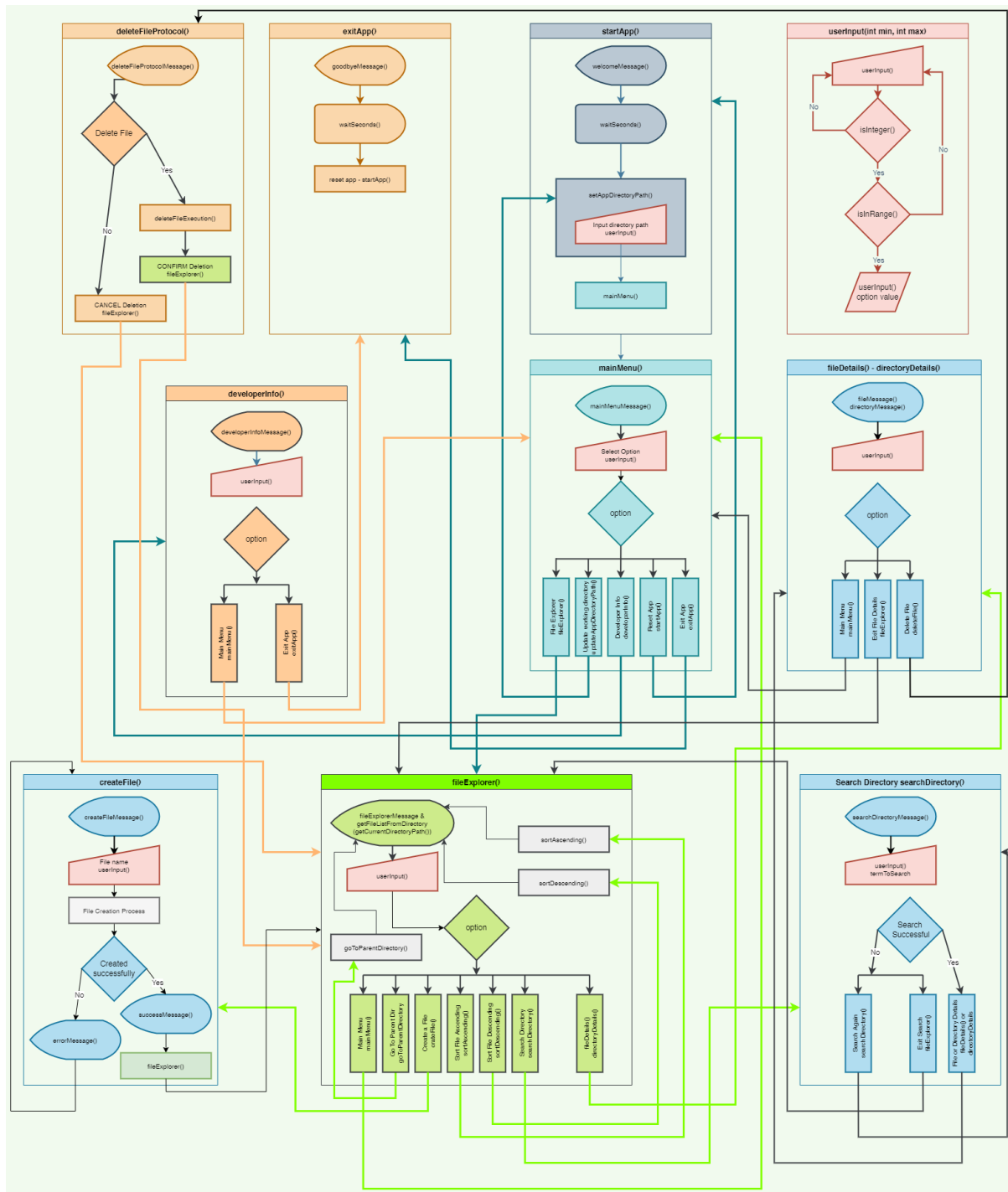
In the second part of our 4th sprite we will develop `sortAscending()` and `sortDescending()` methods which are going to sort the file and then present it using the `fileExplorer()`. Those changes are going to influence the whole code and many changes in the code will need to be done to be able to join the soring functionality with the all existing functionality implemented in the `fileExplorer()`.

In the third part of the 4th sprint we will implement the `earchFolder()` method to give the user an option to search for a file or folder in the `fileExplorer()` after this is done app needs to be tested and documented.

Sprite 5:

In our 5th and last sprite we will test the app do necessary changes and refactoring to make the code shorter where it can be done easily. After all the testing is done we will add more comments in the code and then we will make the necessary documentation for this app to be able to make a nice report, app flow chart and algorithm details.

App Flow Chart



App sample snips

Welcome and Setting Absolute Working Directory path

```
+-----[ Welcome ]-----+
|
| Application Developer : Drazen Drinic
| Application Name: Virtual Key For Your Repositories
| Application GitHub: https://github.com/Drazen-BBG/Simplilearn\_OOPS\_Project
|
+-----[ Next steps are below ]-----+

+-----[ Setting Absolute Working Directory Path ]-----+
|
| Step 1 > Create a folder on your PC and give it one word name
| Step 2 > Open the folder and click on the folder icon in the address bar
| Step 3 > Copy the address link so you can past it in the next step
|
+-----[ Now we need to past or enter the path down below ]-----+

Please paste or enter your absolute working directory path here: F:\IntelliJ_Projects\Simplilearn_OOPS_Project\test_app_path|
```

Main menu

```
Please paste or enter your absolute working directory path here: F:\IntelliJ_Projects\Simplilearn_OOPS_Project\test_app_path

----->>> Thanks for setting the absolute working directory path <<<-----
----->>> The path is: F:\IntelliJ_Projects\Simplilearn_OOPS_Project\test_app_path <<<-----

+-----[ Main Menu ]-----+
|
| [1] File Explorer
| [2] Update working directory path
| [3] Developer Info
| [4] Restart App
| [5] Exit App
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here: |
```

Updating Absolute Working Directory

```
Please enter your choice here: 2

----->>> You entered: > 2 < which is a valid option <<<-----
----->>> Thanks For Your Input <<<-----

+-----[ Updating Absolute Working Directory Path ]-----+
|
| Step 1 > Find a folder on your PC you want to make app directory
| Step 2 > Open the folder and click on the folder icon in the address bar
| Step 3 > Copy the address link so you can past it in the next step
|
+-----[ Now we need to past or enter the path down below ]-----+

Please paste or enter your absolute working directory path here: F:\IntelliJ_Projects\Simplilearn_OOPS_Project\test_app_path
```

Developer Info

```
Please enter your choice here: 3

----->>> You entered: > 3 < which is a valid option <<-----
----->>> Thanks For Your Input <<<-----

+-----[ Developer Info ]-----+
|
| Application Developer : Drazen Drinic
| LinkedIn: https://www.linkedin.com/in/drazendrnic/
| Application GitHub: https://github.com/Drazen-BBG/SimpliLearn\_00PS\_Project
|
| [1] Main Menu
| [2] Exit App
|
+-----[ Next steps are below ]-----+

Please enter your choice here:
```

File Explorer

```
Please enter your choice here: 1

----->>> You entered: > 1 < which is a valid option <<-----
----->>> Thanks For Your Input <<<-----

+-----[ File Explorer ]-----+
|
| General File Explorer options:
|
| [1] Go to Main Menu
| [2] Go to parent directory
| [3] Create a file
| [4] Sort file ascending
| [5] Sort file descending
| [6] Search directory
|
| Select a file for details and more options:
|
| [7] abc.txt
| [8] bcd.txt
| [9] cde.txt
| [10] file
| [11] file2
| [12] FileWithNoExtension
| [13] Test_1.txt
| [14] Test_2.txt
| [15] Test_3.txt
| [16] Test_Folder [Folder]
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here: |
```

Sort Descending

```
Please enter your choice here: 4

----->>> You entered: > 4 < which is a valid option <<-----
----->>> Thanks For Your Input <<<-----

+-----[ File Explorer ]-----+
|
| General File Explorer options:
|
| [1] Go to Main Menu
| [2] Go to parent directory
| [3] Create a file
| [4] Sort file ascending
| [5] Sort file descending
| [6] Search directory
|
| Select a file for details and more options:
|
| [7] abc.txt
| [8] bcd.txt
| [9] cde.txt
| [10] file
| [11] file2
| [12] FileTest.txt
| [13] FileWithNoExtension
| [14] Test_1.txt
| [15] Test_2.txt
| [16] Test_3.txt
| [17] Test_Folder [Folder]
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here: |
```

Create a File

```
Please enter your choice here: 3

----->>> You entered: > 3 < which is a valid option <<-----
----->>> Thanks For Your Input <<<-----

+-----[ Create File ]-----+
|
| Step 1 > You will need to know the name for your file and its extension
| Step 2 > Write your file name in this format <fileName.extension>
| Example > MyTextFile.txt
|
+-----[ Please Enter The File Name Below ]-----+

Please enter file name here: FileTest.txt

----->>> Your file: FileTest.txt was created. <<-----
-->>> The file path is: F:\IntelliJ_Projects\SimpliLearn_00PS_Project\test_app_path\FileTest.txt <<---
```


File Details

```
Please enter your choice here: 7

----->>> You entered: > 7 < which is a valid option <<<-----
----->>> Thanks For Your Input <<<-----

+-----[ File details ]-----+
|
| File name: abc.txt
| File path: F:\IntelliJ_Projects\SimpleLearn_00PS_Project\test_app_path\abc.txt
| File size: 15.82905 kilobytes
|
| [1] Go to Main Menu
| [2] Exit file details
| [3] Delete file
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here: |
```

Delete File

```
Please enter your choice here: 3

----->>> You entered: > 3 < which is a valid option <<<-----
----->>> Thanks For Your Input <<<-----

+-----[ Confirm you want to delete ]-----+
|
| File name: file2
| File path: F:\IntelliJ_Projects\SimpleLearn_00PS_Project\test_app_path\file2
| File size: 0.00000
|
| [1] To CONFIRM deletion
| [2] To CANCEL deletion
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here: 1

----->>> You entered: > 1 < which is a valid option <<<-----
----->>> Thanks For Your Input <<<-----

----->>> Your file: file2 was deleted. <<<-----
```

Directory Details

```
Please enter your choice here: 17

----->>> You entered: > 17 < which is a valid option <<<-----
----->>> Thanks For Your Input <<<-----

+-----[ Directory details ]-----+
|
| Directory name: Test_Folder
| Directory path: F:\IntelliJ_Projects\SimpleLearn_00PS_Project\test_app_path\Test_Folder
| Directory size: 185.07510 kilobytes
|
| [1] Go to Main Menu
| [2] Exit directory details
| [3] Enter directory
|
+-----[ Please Enter Your Choice Below ]-----+

Please enter your choice here:
```

Goodbye

Please enter your choice here: 5

----->>> You entered: > 5 < which is a valid option <<-----

----->>> Thanks For Your Input <<-----

+-----[Goodbye]-----+

|

Thanks for trying my app!!!

See you next time!!!

|

+-----***-----+

Process finished with exit code 0

|

Algorithm

```
package com.lockersPvtLtd.application;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import java.util.regex.Pattern;
import java.util.Arrays;

public class VirtualKeyForYourRepositories {

    // Path variables

    static String absoluteAppDirectoryPath; // It is the path provided by the user and can only be
    changed using setAppDirectoryPath() or updateAppDirectoryPath()

    static String currentDirectoryPath; // It is the path which changes as we navigate through
    fileExplorer by entering folders or by going to a parent folder


    // Other global variables

    static String appName = "Virtual Key For Your Repositories";

    static int consoleBoxWidth = 100; //It is the width of the console text box, suggested minimum is
    90

    static String gitHubAppLink = "https://github.com/Drazen-BBG/SimpliLearn_OOPS_Project";
    static String developerName = "Drazen Drinic";
    static String developerLinkedin = "https://www.linkedin.com/in/drazendrnic/";

    // numberingInFileExplorer is reset to 0 and then used to number each option for the user
```

```

static int numberingInFileExplorer = 0;

// loopIterations is numbering inside loops to be able to calculate and make a difference between
General options and file related options

static int loopIterations = 0;


// Getters and Setters

public static String getAbsolutePath() {
    return absoluteAppDirectoryPath;
}

public static void setAbsolutePath(String absoluteAppDirectoryPath) {
    VirtualKeyForYourRepositories.absoluteAppDirectoryPath = absoluteAppDirectoryPath;
}

public static String getCurrentDirectoryPath() {
    return currentDirectoryPath;
}

public static void setCurrentDirectoryPath(String currentDirectoryPath) {
    VirtualKeyForYourRepositories.currentDirectoryPath = currentDirectoryPath;
}

public static String getAppName() {
    return appName;
}

public static void setAppName(String appName) {
    VirtualKeyForYourRepositories.appName = appName;
}

public static int getConsoleBoxWidth() {
    return consoleBoxWidth;
}

public static void setConsoleBoxWidth(int consoleBoxWidth) {
    VirtualKeyForYourRepositories.consoleBoxWidth = consoleBoxWidth;
}

```

```

}

public static String getGitHubAppLink() {
    return gitHubAppLink;
}

public static void setGitHubAppLink(String gitHubAppLink) {
    VirtualKeyForYourRepositories.gitHubAppLink = gitHubAppLink;
}

public static String getDeveloperName() {
    return developerName;
}

public static void setDeveloperName(String developerName) {
    VirtualKeyForYourRepositories.developerName = developerName;
}

public static String getDeveloperLinkedin() {
    return developerLinkedin;
}

public static void setDeveloperLinkedin(String developerLinkedin) {
    VirtualKeyForYourRepositories.developerLinkedin = developerLinkedin;
}

public static int getNumberingInFileExplorer() {
    return numberingInFileExplorer;
}

public static void setNumberingInFileExplorer(int numberingInFileExplorer) {
    VirtualKeyForYourRepositories.numberingInFileExplorer = numberingInFileExplorer;
}

public static int getLoopIterations() {
    return loopIterations;
}

public static void setLoopIterations(int loopIterations) {
    VirtualKeyForYourRepositories.loopIterations = loopIterations;
}

```

```
// main()

public static void main(String[] args) {
    startApp();
}
```

```
// startApp()

private static void startApp(){
    //Clear the console code, having issues to make it work in IntelliJ IDEA
    System.out.print("\033[H\033[2J");
    System.out.flush();

    //Running the app
    welcomeMessage();
    waitSeconds(2);
    setAppDirectoryPath();
    mainMenu();
}

// waitSeconds() delays execution of the code in seconds entered as parameter.
private static void waitSeconds(int waitSeconds){

    int milliSeconds = 1000 * waitSeconds;

    try {
        Thread.sleep( milliSeconds);
    } catch (InterruptedException ie) {
```

```
        Thread.currentThread().interrupt();
    }
}

// exitApp()
private static void exitApp(){
    goodbyeMessage();
    waitSeconds(3);
    System.exit(0);
}
```

```
// lineBuilder methods help us to build lines in the console, for better presentation.
private static String lineBuilderCenterAligned(int lineLength, String mainString, char
firstAndLastPlaceHolder, char mainPlaceHolder){
    String line = "";
    StringBuilder prefix = new StringBuilder();
    StringBuilder postfix = new StringBuilder();

    int prefixLength = (lineLength - mainString.length())/2;
    int postfixLength = lineLength - mainString.length() - prefixLength;

    for (int i = 0; i < prefixLength; i++){
        prefix.append(mainPlaceHolder);
    }
    prefix.insert(0, firstAndLastPlaceHolder);

    for (int i = 0; i < postfixLength; i++){
        postfix.append(mainPlaceHolder);
    }
}
```

```

        postfix.append(firstAndLastPlaceholder);

        line = prefix + mainString + postfix;

        return line;
    }

    private static String lineBuilderLeftAligned(int lineLength, String mainString, char
firstAndLastPlaceholder, char mainPlaceholder){
        String line;
        StringBuilder prefix = new StringBuilder();
        StringBuilder postfix = new StringBuilder();

        int prefixLength = 1;
        int postfixLength = lineLength - mainString.length() - prefixLength;

        for (int i = 0; i < prefixLength; i++){
            prefix.append(mainPlaceholder);
        }
        prefix.insert(0, firstAndLastPlaceholder);

        for (int i = 0; i < postfixLength; i++){
            postfix.append(mainPlaceholder);
        }
        postfix.append(firstAndLastPlaceholder);

        line = prefix + mainString + postfix;

        return line;
    }

```


// welcomeMessage() is the first method to call, it greets the user.

```
private static void welcomeMessage(){
    String firstLine = "[ Welcome ]";
    String lastLine = "[ Next steps are below ]";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add("Application Developer : " + getDeveloperName());
    mainMessageContentList.add("Application Name: " + getAppName());
    mainMessageContentList.add("Application GitHub: " + getGitHubAppLink());

    buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
}
```

//goodbyeMessage() is called when we want to exit the program

```
private static void goodbyeMessage(){
    String firstLine = "[ Goodbye ]";
    String lastLine = "***";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add("Thanks for trying my app!!!");
    mainMessageContentList.add("See you next time!!!");

    buildMessageContent(firstLine, mainMessageContentList, lastLine, true);
}
```

```
private static void developerInfoMessage(){
```

```

String firstLine = "[ Developer Info ]";

String userInputLine = "Please enter your choice here: ";

String lastLine = "[ Next steps are below ]";


List<String> mainMessageContentList = new ArrayList<>();
mainMessageContentList.add("Application Developer : " + getDeveloperName());
mainMessageContentList.add("LinkedIn: " + getDeveloperLinkedin());
mainMessageContentList.add("Application GitHub: " + getGitHubAppLink());
mainMessageContentList.add("");
mainMessageContentList.add("[1] Main Menu");
mainMessageContentList.add("[2] Exit App");


buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
buildUserInputLine(userInputLine);
}

// developerInfo() displays developer info
private static void developerInfo(){

    developerInfoMessage();


    int min = 1;
    int max = 2;
    int option;

    //Getting validated option from user
    option = userInput(min, max);


    switch (option){
        case 1:
            mainMenu();
            break;
        case 2:

```

```

        exitApp();
        break;
    }
}

```

// setAppDirectoryPathMessage() is a method used to separate message from functionality in the setAppDirectoryPath()

```

private static void setAppDirectoryPathMessage(){
    String firstLine = "[ Setting Absolute Working Directory Path ]";
    String userInputLine = "Please paste or enter your absolute working directory path here: ";
    String lastLine = "[ Now we need to past or enter the path down below ]";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add(" Step 1 > Create a folder on your PC and give it one word name");
    mainMessageContentList.add(" Step 2 > Open the folder and click on the folder icon in the address bar ");
    mainMessageContentList.add(" Step 3 > Copy the address link so you can past it in the next step");

    buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
    buildUserInputLine(userInputLine);
}

// setAppDirectoryPath() sets the path for the app folder at the beginning of the program
private static void setAppDirectoryPath() {

```

```

    setAppDirectoryPathMessage();

```

```

    Scanner scanner = new Scanner(System.in);

```

```

    String tempPath =scanner.next();

```

```

//scanner.close(); This was causing issues

System.out.println("");

if(Pattern.matches("(\\V)+|(\\\\\\\\)+", tempPath)) {

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Seems your input
was invalid <<<", '+', '-'));

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Check the
instructions below and try again <<<", '+', '-'));

    setAppDirectoryPath();

}else {

    if (new File(tempPath).exists() && new File(tempPath).isDirectory()){

        setCurrentDirectoryPath(tempPath);

        setAbsoluteAppDirectoryPath(tempPath);

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Thanks for setting
the absolute working directory path <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> The path is: " +
tempPath + " <<<", '-', '-'));

    }else{

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Seems your input
was invalid <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Check the
instructions below and try again <<<", '-', '-'));

        setAppDirectoryPath();

    }

}

}

// updateAppDirectoryPathMessage() is a method used to separate message from functionality in
the updateAppDirectoryPath()

private static void updateAppDirectoryPathMessage(){

    String firstLine = "[ Updating Absolute Working Directory Path ]";

    String userInputLine = "Please paste or enter your absolute working directory path here: ";

    String lastLine = "[ Now we need to past or enter the path down below ]";

```

```

List<String> mainMessageContentList = new ArrayList<>();

mainMessageContentList.add(" Step 1 > Find a folder on your PC you want to make app
directory");

mainMessageContentList.add(" Step 2 > Open the folder and click on the folder icon in the
address bar ");

mainMessageContentList.add(" Step 3 > Copy the address link so you can past it in the next
step");


buildMessageContent(firstLine, mainMessageContentList, lastLine, false);

buildUserInputLine(userInputLine);

}

// updateAppDirectoryPath() is a method that gives user an option to change the
absoluteAppDirectoryPath, and therefore it also updates the currentDirectoryPath

private static void updateAppDirectoryPath() {

    updateAppDirectoryPathMessage();

    Scanner scanner = new Scanner(System.in);

    String tempPath = scanner.next();

    //scanner.close(); This was causing issues

    System.out.println("");

    if(Pattern.matches("(\\V)+|(\\\\\\\\)+", tempPath)) {

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Seams your input
was invalid <<<", '+', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Check the
instructions below and try again <<<", '+', '-'));

        setAppDirectoryPath();

    }else {

        if (new File(tempPath).exists() && new File(tempPath).isDirectory()){

```

```

        setCurrentDirectoryPath(tempPath);

        setAbsoluteAppDirectoryPath(tempPath);

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Thanks for
updating the absolute working directory path <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> The path is: " +
tempPath + " <<<", '-', '-'));

    }else{

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Seems your input
was invalid <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Check the
instructions below and try again <<<", '-', '-'));

        setAppDirectoryPath();

    }

}

}

```

```

// User input methods to take input from user and to validate the input

// For this to work every other method calling those methods have to have min and max values

// userInput() is called in every other method when we need a user input, other related methods
below are helper methods to make it all work

private static int userInput(int min, int max){

```

```

    Scanner scanner = new Scanner(System.in);

    String userInputString = scanner.next();

```

```

    int userInputInteger;

    int userInputIntegerNextTry;

```

```

    if (isInteger(userInputString)){

        userInputInteger = Integer.parseInt(userInputString);
    }

```

```

    if (isInRange(userInputInteger, min, max)){

        // Message for valid input

        System.out.println("");

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> You entered: > "
+ userInputInteger + " < which is a valid option <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Thanks For Your
Input <<<", '-', '-'));

        //System.out.println("");

        return userInputInteger;

    }else{

        // Wrong input NOT IN RANGE

        System.out.println("");

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> You entered: > "
+ userInputInteger + " < which is in not in range between " + min+"-"+max + " <<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Please try again
<<<", '-', '-'));

        System.out.println("");

        System.out.print("Please enter your choice here: ");

        // It would be great to print the appropriate menu again

        userInputIntegerNextTry = userInput(min, max);

        return userInputIntegerNextTry;

    }

}

}else{

    // Wrong input NOT AN INTEGER

    System.out.println("");

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> You entered: > " +
userInputString + " < which is NOT a WHOLE number <<<", '-', '-'));

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Please try again
<<<", '-', '-'));

    System.out.println("");

    System.out.print("Please enter your choice here: ");

```

```

        userInputIntegerNextTry = userInput(min, max);
        return userInputIntegerNextTry;
    }
}

```

```

// isInteger() checks if the user input is integer
private static boolean isInteger(String stringToCheck){
    if (stringToCheck == null){
        return false;
    }
    try {
        int integerNumber = Integer.parseInt(stringToCheck);
    }catch (NumberFormatException nfe){
        return false;
    }
    return true;
}

```

```

// isInRange() method checks if the user input is in range of available options
private static boolean isInRange(int numberToTest, int min, int max){
    if (numberToTest >= min && numberToTest <= max){
        return true;
    }else{
        return false;
    }
}

```

```

// Message builder methods

```

```

// buildFirstLine()

```



```

private static void buildFirstLine(String firstLine){
    System.out.println();

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), firstLine, '+', '-'));

    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), "", '|', ' '));
}

// buildMessageContent()

private static void buildMessageContent(List mainMessageContentList, Boolean isCentered){
    if (isCentered){
        for (int i = 0; i<mainMessageContentList.size(); i++){
            System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(),
mainMessageContentList.get(i).toString(), '|', ' '));
        }
    }else {
        for (int i = 0; i<mainMessageContentList.size(); i++){
            System.out.println(lineBuilderLeftAligned(getConsoleBoxWidth(),
mainMessageContentList.get(i).toString(), '|', ' '));
        }
    }
}

// buildMessageContent() with buildFirstLine() and buildLastLine() included

private static void buildMessageContent(String firstLine, List mainMessageContentList, String
lastLine, Boolean isCentered){
    buildFirstLine(firstLine);

    if (isCentered){
        for (int i = 0; i<mainMessageContentList.size(); i++){
            System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(),
mainMessageContentList.get(i).toString(), '|', ' '));
        }
    }else {
        for (int i = 0; i<mainMessageContentList.size(); i++){
            System.out.println(lineBuilderLeftAligned(getConsoleBoxWidth(),
mainMessageContentList.get(i).toString(), '|', ' '));
        }
    }
}

```

```

    }
}

    buildLastLine(lastLine);
}
// buildLastLine()
private static void buildLastLine(String lastLine){
    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), "", '|', ' '));
    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), lastLine, '+', '-'));
    System.out.println();
}
// buildUserInputLine()
private static void buildUserInputLine(String userInputLine){
    System.out.print(userInputLine);
}

```

// mainMenuMessage() is a method used to separate message from functionality in the mainMenu()

```

private static void mainMenuMessage(){

    //creating the console look
    String firstLine = "[ Main Menu ]";
    String userInputLine = "Please enter your choice here: ";
    String lastLine = "[ Please Enter Your Choice Below ]";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add("[1] File Explorer");
    mainMessageContentList.add("[2] Update working directory path");
    mainMessageContentList.add("[3] Developer Info");
}

```

```

mainMessageContentList.add("[4] Restart App");
mainMessageContentList.add("[5] Exit App");

buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
buildUserInputLine(userInputLine);

}

// mainMenu() method is the main menu presented to the user
private static void mainMenu(){

    mainMenuMessage();

    // Setting min and max range
    int min = 1;
    int max = 5;
    int option;

    //Getting validated option from user
    option = userInput(min, max);

    switch (option){
        case 1:
            setCurrentDirectoryPath(getAbsoluteAppDirectoryPath());
            fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
            break;
        case 2:
            updateAppDirectoryPath();
            mainMenu();
            break;
        case 3:
            developerInfo();

```

```

        break;
    case 4:
        startApp();
        break;
    case 5:
        exitApp();
        break;
    }
}

```

// fileExplorerMessage() is a method used to separate message from functionality in the fileExplorer()

```

private static void fileExplorerMessage(File[] inputFileListToExplore){

    //creating the console look
    String firstLine = "[ File Explorer ]";
    String userInputLine = "Please enter your choice here: ";
    String lastLine = "[ Please Enter Your Choice Below ]";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add("General File Explorer options:");
    mainMessageContentList.add("");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Go to Main Menu");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Go to parent directory");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Create a file");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Sort file ascending");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Sort file descending");
    mainMessageContentList.add("[ " + (++numberingInFileExplorer) + "]" + " Search directory");
}

```

```

mainMessageContentList.add("");

mainMessageContentList.add("Select a file for details and more options:");

mainMessageContentList.add("");

buildFirstLine(firstLine);

buildMessageContent(mainMessageContentList, false);

if (inputFileListToExplore != null && inputFileListToExplore.length > 0){
    for (File file : inputFileListToExplore){
        ++loopIterations;

        // Check if the file is a directory
        if (file.isDirectory()){
            System.out.println(lineBuilderLeftAligned(getConsoleBoxWidth(),
"["+++numberingInFileExplorer+"]" + " " + file.getName() + " [Folder]", '|', ' '));
        }else {
            System.out.println(lineBuilderLeftAligned(getConsoleBoxWidth(),
"["+++numberingInFileExplorer+"]" + " " + file.getName(), '|', ' '));
        }
    }
}

buildLastLine(lastLine);

buildUserInputLine(userInputLine);
}

// fileExplorer() presents content of a folder and gives General and file specific options to the user
private static void fileExplorer(File[] inputFileListToExplore){
    //Restarting the numberingInFileExplorer and loopIterations on each fileExplorer() call
    setNumberingInFileExplorer(0);
    setLoopIterations(0);

    fileExplorerMessage(inputFileListToExplore);
}

```

```
// Setting min and max range

int min = 1;

int max = getNumberingInFileExplorer();

int option;

option = userInput(min, max);


//Switch General menu
switch (option){
    case 1:
        mainMenu();
        break;
    case 2:
        goToParentDirectory();
        break;
    case 3:
        crateFile();
        waitSeconds(2);
        fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
        break;
    case 4:
        fileExplorer(sortAscending(getFileListFromDirectory(getCurrentDirectoryPath())));
        break;
    case 5:
        fileExplorer(sortDescending(getFileListFromDirectory(getCurrentDirectoryPath())));
        break;
    case 6:
        searchDirectory(getFileListFromDirectory(getCurrentDirectoryPath()));
        break;
}


//Files and Folders Menu control down below
```

```

if (option>6 && option<=getNumberingInFileExplorer()){
    File fileToProcess = inputFileListToExplore[(option-(getNumberingInFileExplorer()-
getLoopIterations()))-1];
    if (fileToProcess.isDirectory()){
        // If option selected is a folder
        // setCurrentDirectoryPath
        setCurrentDirectoryPath(fileToProcess.getPath());
        directoryDetails(fileToProcess);
    }else {
        // If bot a folder it must be than a file so let's see its details using the fileDetails() method
        fileDetails(fileToProcess);
    }
}
}
}

```

```

// fileDetailsMessage() is a method used to separate message from functionality in the fileDetails()
private static void fileDetailsMessage(File fileToProcess, double fileSizeInKiloBytes){
    //creating the console look
    String firstLine = "[ File details ]";
    String userInputLine = "Please enter your choice here: ";
    String lastLine = "[ Please Enter Your Choice Below ]";

    List<String> mainMessageContentList = new ArrayList<>();
    mainMessageContentList.add("File name: " + fileToProcess.getName());
    mainMessageContentList.add("File path: " + fileToProcess.getAbsolutePath());
    mainMessageContentList.add("File size: " + String.format("%.5f", fileSizeInKiloBytes) + "
kilobytes");
    mainMessageContentList.add("");
    mainMessageContentList.add("[1] + " Go to Main Menu");
    mainMessageContentList.add("[2] + " Exit file details");
}

```

```

mainMessageContentList.add("[3]" + " Delete file");

buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
buildUserInputLine(userInputLine);
}

// fileDetails() lists the details of a selected file and offers an option to delete the file
private static void fileDetails(File fileToProcess){
    Path fileToProcessPath = Paths.get(fileToProcess.getAbsolutePath());
    double fileSize = 0;
    try {
        fileSize = Files.size(fileToProcessPath);
    }catch (IOException e){
        e.printStackTrace();
        //Make a code to try again
    }
    double fileSizeInKiloBytes = fileSize/2024;

    fileDetailsMessage(fileToProcess, fileSizeInKiloBytes);

    // Setting min and max range
    int min = 1;
    int max = 3;
    int option;
    //Getting validated option from user
    option = userInput(min, max);

    switch (option) {
        case 1:
            setCurrentDirectoryPath(getAbsoluteAppDirectoryPath());
            mainMenu();
            break;

```


case 2:

```
fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
```

```
break;
```

case 3:

```
deleteFileProtocol(fileToProcess);
```

```
waitSeconds(2);
```

```
fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
```

```
break;
```

```
}
```

```
}
```

// directoryDetailsMessage() is a method used to separate message from functionality in the directoryDetails()

```
private static void directoryDetailsMessage(File fileToProcess, double fileSizeInKiloBytes){
```

```
//creating the console look
```

```
String firstLine = "[ Directory details ]";
```

```
String userInputLine = "Please enter your choice here: ";
```

```
String lastLine = "[ Please Enter Your Choice Below ]";
```

```
List<String> mainMessageContentList = new ArrayList<>();
```

```
mainMessageContentList.add("Directory name: " + fileToProcess.getName());
```

```
mainMessageContentList.add("Directory path: " + fileToProcess.getAbsolutePath());
```

```
mainMessageContentList.add("Directory size: " + String.format("%.5f", fileSizeInKiloBytes) + " kilobytes");
```

```
mainMessageContentList.add("");
```

```
mainMessageContentList.add("[1]" + " Go to Main Menu");
```

```
mainMessageContentList.add("[2]" + " Exit directory details");
```

```
mainMessageContentList.add("[3]" + " Enter directory");
```

```
buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
```

```
buildUserInputLine(userInputLine);
```

```
}
```

// directoryDetails() lists the details of a selected directory and offers an option to enter the directory

```
private static void directoryDetails(File directoryToProcess){  
    Path fileToProcessPath = Paths.get(directoryToProcess.getAbsolutePath());  
    double directorySize = 0;  
  
    File[] filesInDirectory = directoryToProcess.listFiles();  
    int numberOfFilesInDirectory = filesInDirectory.length;  
  
    for (int i = 0; i < numberOfFilesInDirectory; i++){  
        if (filesInDirectory[i].isFile()){  
            directorySize += filesInDirectory[i].length();  
        }else{  
            directoryDetails(directoryToProcess);  
        }  
    }  
  
    double fileSizeInKiloBytes = directorySize/2024;  
  
    directoryDetailsMessage(directoryToProcess, fileSizeInKiloBytes);  
  
    // Setting min and max range  
    int min = 1;  
    int max = 3;  
    int option;  
    //Getting validated option from user  
    option = userInput(min, max);  
  
    switch (option) {  
        case 1:  
            mainMenu();  
    }
```

```

        break;
    case 2:
        goToParentDirectory();
        break;
    case 3:
        goIntoDirectory();
    }
}

// goToParentDirectory()
private static void goToParentDirectory(){
    Path currentDirectoryPath = Paths.get(getCurrentDirectoryPath());
    Path parentPath = currentDirectoryPath.getParent();
    setCurrentDirectoryPath(parentPath.toString());
    fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
}

// goIntoDirectory()
private static void goIntoDirectory(){
    //We need to change the currentDirectoryPath before calling this method, since this method is
    just calling the fileExplorer
    fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
}

```

// createFileMessage() is a method used to separate message from functionality in the createFile()

```

private static void createFileMessage(){
    //creating the console look
    String firstLine = "[ Create File ]";
    String userInputLine = "Please enter file name here: ";
}

```

```

String lastLine = "[ Please Enter The File Name Below ]";

List<String> mainMessageContentList = new ArrayList<>();

mainMessageContentList.add(" Step 1 > You will need to know the name for your file and its
extension");

mainMessageContentList.add(" Step 2 > Write your file name in this format
<fileName.extension>");

mainMessageContentList.add(" Example > MyTextFile.txt");

buildMessageContent(firstLine, mainMessageContentList, lastLine, false);

buildUserInputLine(userInputLine);
}

// createFile() creates a new file in the folder we were seeing using fileExplorer()
private static void createFile(){

    createFileMessage();

    //User input
    Scanner scanner = new Scanner(System.in);
    String fileName =scanner.next();

    try {
        File file = new File(getCurrentDirectoryPath(), fileName);
        if(!file.exists()) {
            if(file.createNewFile()) {
                System.out.println();
                System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Your file: " +
file.getName() + " was created. <<<", '-', '-'));
                System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> The file path is:
" + file.getAbsolutePath() + " <<<", '-', '-'));
                waitSeconds(2);
                fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
            }
        }
    }
}

```



```

mainMessageContentList.add("File path: " + fileToProcess.getAbsolutePath());
mainMessageContentList.add("File size: " + String.format("%.5f", fileSizeInKiloBytes));
mainMessageContentList.add("");
mainMessageContentList.add("[1] To CONFIRM deletion");
mainMessageContentList.add("[2] To CANCEL deletion");

buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
buildUserInputLine(userInputLine);
}

```

// deleteFileProtocol() organizes the whole thing about deleting the file using deleteFileProtocolMessage() and deleteFileExecution()

```

private static void deleteFileProtocol(File fileToProcess){
    Path fileToProcessPath = Paths.get(fileToProcess.getAbsolutePath());
    double fileSize = 0;
    try {
        fileSize = Files.size(fileToProcessPath);
    }catch (IOException e){
        e.printStackTrace();
        //Make a code to try again
    }
    double fileSizeInKiloBytes = fileSize/2024;

```

```

deleteFileProtocolMessage(fileToProcess, fileSizeInKiloBytes);

```

// Setting min and max range

```

int min = 1;
int max = 2;
int option;

```

//Getting validated option from user

```
option = userInput(min, max);
```

```
switch (option) {
```

```
    case 1:
```

```
        deleteFileExecution(fileToProcess);
```

```
        break;
```

```
    case 2:
```

```
        fileDetails(fileToProcess);
```

```
}
```

```
}
```

```
// deleteFileExecution() deletes the actual file the
```

```
private static void deleteFileExecution(File fileToProcess){
```

```
    File fileToDelete = new File(fileToProcess.getAbsolutePath());
```

```
    if (fileToDelete.delete()){
```

```
        System.out.println();
```

```
        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Your file: " +  
fileToProcess.getName() + " was deleted. <<<", '-', '-'));
```

```
    }else {
```

```
        System.out.println();
```

```
        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Your file: " +  
fileToProcess.getName() + " was NOT created. <<<", '-', '-'));
```

```
    }
```

```
}
```

```
// getFileListFromFolder lists files and folders from the currentDirectoryPath and returns an array  
of Files
```

```
private static File[] getFileListFromDirectory(String pathToFolder){
```

```
    File directoryPath = new File(pathToFolder);
```

```

    File fileList[] = directoryPath.listFiles();

    return fileList;
}

// sortAscending() sorts files it gets from getFileListFromFolder() to ascending order
private static File[] sortAscending(File[] fileListToSort){

    Arrays.sort(fileListToSort);

    return fileListToSort;
}

// sortDescending() sorts files it gets from getFileListFromFolder() to descending order
private static File[] sortDescending(File[] fileListToSort){

    Arrays.sort(fileListToSort, Collections.reverseOrder());

    return fileListToSort;
}

// searchFolderMessage() is a method used to separate message from functionality in the
searchFolder

private static void searchDirectoryMessage(){

    String firstLine = "[ Search Folder ]";

    String userInputLine = "Please enter the term to search: ";

    String lastLine = "[ Please Enter The Search Term Below ]";

    List<String> mainMessageContentList = new ArrayList<>();

    mainMessageContentList.add("Rule #1 This search will find only exact match for your search");

    mainMessageContentList.add("Rule #2 This search is not case sensitive");

    mainMessageContentList.add("Rule #3 If your file has extension you need to include it in your
search");

    mainMessageContentList.add("Example: abc.txt & ABC.TXT are both valid search terms for a file
abc.txt");

    buildMessageContent(firstLine, mainMessageContentList, lastLine, false);

    buildUserInputLine(userInputLine);

```



```

}

// searchFolder() gives user an option to check if a specific file or folder name exists in the
// directory and if it exists it enters the folder, or it shows file details

private static void searchDirectory(File[] fileListToSearch){

    searchDirectoryMessage();

    Scanner scanner = new Scanner(System.in);

    String termToSearch = scanner.next();

    boolean foundIt = false;

    if (fileListToSearch == null){

        System.out.println();

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> The folder is empty
<<<", '-', '-'));

        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Let's go back to the
file explorer <<<", '-', '-'));

        waitSeconds(2);

        fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));

    }else{

        for (int i = 0; i < fileListToSearch.length; i++) {

            String fileName = fileListToSearch[i].getName();

            if (fileName.equalsIgnoreCase(termToSearch)){

                // code if file found and change foundIt to true

                System.out.println();

                if (fileListToSearch[i].isDirectory()){

                    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> You found a
directory named " + termToSearch + " <<<", '-', '-'));

                    System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Lets see the
details <<<", '-', '-'));

                }else {

```

```
        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> You found a  
file named " + termToSearch + " <<<", '-', '-'));
```

```
        System.out.println(lineBuilderCenterAligned(getConsoleBoxWidth(), ">>> Lets see the  
details <<<", '-', '-'));
```

```
    }
```

```
        if (fileListToSearch[i].isDirectory()){
```

```
            setCurrentDirectoryPath(fileListToSearch[i].getPath()); // this helps when we exit the  
details to end up in the same directory as when we did the search
```

```
            directoryDetails(fileListToSearch[i]);
```

```
        }else {
```

```
            fileDetails(fileListToSearch[i]);
```

```
        }
```

```
        foundIt = true;
```

```
    }
```

```
}
```

```
}
```

```
if (foundIt == false){
```

```
    String firstLine = "[ Search Result NOT Found ]";
```

```
    String userInputLine = "Please enter your choice here: ";
```

```
    String lastLine = "[ Please Enter The Search Term Below ]";
```

```
List<String> mainMessageContentList = new ArrayList<>();
```

```
mainMessageContentList.add("Seems there is no file or folder named: " + termToSearch);
```

```
mainMessageContentList.add("");
```

```
mainMessageContentList.add("[1] Search again");
```

```
mainMessageContentList.add("[2] Exit Search ");
```

```
buildMessageContent(firstLine, mainMessageContentList, lastLine, false);
```

```
buildUserInputLine(userInputLine);
```

```
// Setting min and max range
```

```

int min = 1;

int max = 2;

int option;

//Getting validated option from user successful
option = userInput(min, max);

switch (option) {
    case 1:
        searchDirectory(getFileListFromDirectory(getCurrentDirectoryPath()));
        break;
    case 2:
        fileExplorer(getFileListFromDirectory(getCurrentDirectoryPath()));
        break;
}
}
}

```

```

//Old methods not in use anymore
private static void goodbyeOldMethod(){
    // Initial setup to make the Goodbye box
    String firstLine = "-----[Goodbye]-----";
    String thankYouLine = "Thanks for trying my app!!!";
    String seeYouLine = "See you next time!!!";
    String lastLine = "-----*****-----";
    int boxWidth = firstLine.length();
}

```

```
// Down below is the code for center align the thankYouLine

int indentBefore1;

int indentAfter1;

if ((thankYouLine.length()%2 == 0){

    indentBefore1 = (((boxWidth+thankYouLine.length())/2) - 1);

    indentAfter1 = (((boxWidth-thankYouLine.length())/2)-1);

}else {

    indentBefore1 = (((boxWidth+thankYouLine.length())/2) - 1);

    indentAfter1 = ((boxWidth-thankYouLine.length())/2) - 2;

}
```

```
// Down below is the code for center align the seeYouLine

int indentBefore2;

int indentAfter2;

if ((seeYouLine.length()%2 == 0){

    indentBefore2 = (((boxWidth+seeYouLine.length())/2) - 1);

    indentAfter2 = (((boxWidth-seeYouLine.length())/2)-1);

}else {

    indentBefore2 = (((boxWidth+seeYouLine.length())/2) - 1);

    indentAfter2 = ((boxWidth-seeYouLine.length())/2) - 2;

}
```

```
// Those are the print statements to make the Goodbye box

System.out.printf("\n+%" + firstLine.length() + "s+ \n", firstLine);

System.out.printf(" | %-"+(firstLine.length()-2)+"s | \n", "");
```

```
String formatString1 =" | %" + indentBefore1 + "s" + " %" + indentAfter1 + "s | ";

System.out.println(String.format(formatString1, thankYouLine, ""));
```

```
String formatString2 =" | %" + indentBefore2 + "s" + " %" + indentAfter2 + "s | ";

System.out.println(String.format(formatString2, seeYouLine, ""));
```

```

System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "");

System.out.printf("%%" + firstLine.length() + "s+ \n", lastLine);


//System.out.println(String.format(" | %20s %20s |", "Thanks", ""));
}

private static void mainMenuOldMethod(){

    String firstLine = "-----[Main Menu]-----";

    String developerLine = "Application Developer : " + developerName;

    String applicationLine = "Application Name: " + appName;

    String lastLine = "-----*****-----";


    System.out.printf("\n%" + firstLine.length() + "s+ \n", firstLine);

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "");


    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "[1] File Explorer");

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "[2] List of Files");

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "[3] Actions for Files");

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "[4] Update the Folder Path");

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "[5] Exit");

    System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "");


    //System.out.printf(" | %-" + (firstLine.length() - 2) + "s | \n", "");

    //System.out.printf(" | %-" + (firstLine.length() - 2) + "s |", "Please enter your option: ");

    //System.out.printf("\n | %-" + (firstLine.length() - 2) + "s | \n", "");


    System.out.printf("%%" + firstLine.length() + "s+ \n", lastLine);

    System.out.print("Please enter your option: ");


}

```

```

private static void filesExplorerOld(){

    String firstLine = "[ File Explorer ]";

    String userInputLine = "Please enter your choice here: ";

    String lastLine = "[ Please Enter Your Choice Below ]";


    int numbering = 0;


    System.out.println();

    System.out.println(lineBuilderCenterAligned(consoleBoxWidth, firstLine, '+', '-'));

    System.out.println(lineBuilderCenterAligned(consoleBoxWidth, "", '|', ' '));


    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "General File Explorer options:", '|',
' '));

    System.out.println(lineBuilderCenterAligned(consoleBoxWidth, "", '|', ' '));


    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Go to
Main Menu", '|', ' '));

    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Create a
file", '|', ' '));

    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Sort file
ascending", '|', ' '));

    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Sort file
descending", '|', ' '));

    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Search
folder", '|', ' '));

    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(++numbering)+"]" + " Exit
App", '|', ' '));


    System.out.println(lineBuilderCenterAligned(consoleBoxWidth, "", '|', ' '));


    File directoryPath = new File(absoluteAppDirectoryPath);

    File filesList[] = directoryPath.listFiles();

```

```
System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "Select a file for details and more options:", '|', ' '));
```

```
System.out.println(lineBuilderCenterAligned(consoleBoxWidth, "", '|', ' '));
```

```
int loopRepetitions = 0;
```

```
for (int i = 0; i < filesList.length; i++){
```

```
    numbering++;
```

```
    //System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(i+1)+"] "+ filesList[i].getName() + " --> " + filesList[i].getAbsolutePath(), '|', ' '));
```

```
    System.out.println(lineBuilderLeftAligned(consoleBoxWidth, "["+(numbering)+"] "+ filesList[i].getName() + "", '|', ' '));
```

```
    loopRepetitions++;
```

```
}
```

```
System.out.println(lineBuilderCenterAligned(consoleBoxWidth, "", '|', ' '));
```

```
System.out.println(lineBuilderCenterAligned(consoleBoxWidth, lastLine, '+', '-')); 
```

```
System.out.println();
```

```
//System.out.println(userInputLine); // Input in next line
```

```
System.out.print(userInputLine); // Input in same line
```

```
}
```

```
private static void fileDetailsOld(String path, int index){
```

```
    File directoryPath = new File(absoluteAppDirectoryPath);
```

```
    File filesList[] = directoryPath.listFiles();
```

```
//need index
```

```
System.out.println(lineBuilderLeftAligned(consoleBoxWidth, filesList[index].getName() + "", '|', ' '));
```

```
System.out.println(lineBuilderLeftAligned(consoleBoxWidth, filesList[index].getAbsolutePath() + "", '|', ' '));
```

```
        System.out.println(lineBuilderLeftAligned(consoleBoxWidth, fileList[index].getAbsolutePath() +
        "", '|', ' '));
```

```
    }
```

```
    private static int calculateIndexOldStuff(int option, int numbering, int loopRepetitions){
```

```
        int indexToReturn;
```

```
        indexToReturn = (option - (numbering - loopRepetitions))-1;
```

```
        return indexToReturn;
```

```
    }
```

```
// Not in use anymore
```

```
private static String getUserInput(){
```

```
    String userInputString;
```

```
    Scanner scanner = new Scanner(System.in);
```

```
    userInputString = scanner.next();
```

```
    //scanner.close(); This was causing issues
```

```
    return userInputString;
```

```
}
```

```
private static void deleteDirectoryProtocolMessage(File fileToProcess, double fileSizeInKiloBytes){
```

```
    String firstLine = "[ Confirm you want to delete ]";
```

```
    String userInputLine = "Please enter your choice here: ";
```

```
    String lastLine = "[ Please Enter Your Choice Below ]";
```

```
List<String> mainMessageContentList = new ArrayList<>();
```

```
mainMessageContentList.add("Directory name: " + fileToProcess.getName());
```

```
mainMessageContentList.add("Directory path: " + fileToProcess.getAbsolutePath());
```

```
mainMessageContentList.add("Directory size: " + String.format("%.5f", fileSizeInKiloBytes));
```

```
mainMessageContentList.add("");
```

```
mainMessageContentList.add("[1] To CONFIRM deletion");
```

```
mainMessageContentList.add("[2] To CANCEL deletion");
```



```
        buildMessageContent(firstLine, mainMessageContentList, lastLine, false);  
        buildUserInputLine(userInputLine);  
    }  
}
```

Conclusion

Application is prepared according to required features and tested. App project directory contains all the needed and extra documentation including a report, app use snips and workflow charts.