Project Specification Document

Virtual Keys Repository Application Prototype

End of Phase 1- OOPS Using Java Data Structures

Student: Thirumavalaven

vthiru97@gmail.com

Full Stack Java Developer

Master's Program

Table of Contents

Virtual Keys Repository Prototype Application

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.

Developer Details

The project is developed by Thirumavalaven. I worked as a Service Delivery Specialist at IBM India Pvt. Ltd.

The code for this project is hosted at https://github.com/Thiru97/Simplilearn2023.git

Project Details

Lockers Pvt. Ltd. aims to digitize their product catalog. For the first phase of the project, they wish to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval, with the goal of delivering a high-end quality product asearly as possible.

Lockers Pvt. Ltd. would like a presentation on the following topics 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 maincontext
    - Option to close the application

Sprint Planning and Tasks Achieved

The project is planned to be completed in 1 sprint. Tasks assumed to be completed inthe 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.

Project Overview

The main objectives of this Projects are

- to gain an understanding of core concepts of the Java Programming Language (abstraction,polymorphism, inheritance, and encapsulation),
- embrace the Eclipse Integrated Development Environment (IDE),
- understand the Agile software development life cycle, and
- Gain familiarity with Java data structures for object-oriented applications.

Implemented Java Concepts

This section will highlight the Java concepts used to create the virtual keys repository applicationprototype. Collections framework, File Handling, Sorting, Flow Control, Exception Handling, Streams API are the core concepts used in this program. ***The entire Application was built in JAVA 16***

Packages

I chose to create a package dedicated to the Locked Me company as per the naming standards.

```
package com.lockedme.virtualkey;
```

Classes, Objects, Methods

I have made all the class as Public for a better code understanding. Some of the methods are made static so that they don't need objects to call them. Below table lists the available class and methods in this application

Classes	Objects	Methods
public class VirtualKey(contains Main Method) *public class MyScanner* *public class Menu* *public class MainMenuOperations* *public class FileOperations*	*sc* *fileList* *myFile*	*public static void setTargetDirectory()* *public static void openScanner()* *public static void closeScanner()* *public static void welcomeScreen()* *public static int mainMenuOptions()* *public static void mainMenu(String targetDirectory)* *public static void businessOptionsMenu(String targetDirectory)* *public static void retrieveFiles(String targetDirectory) throws IOException* *public static void addFile(String targetDirectory) throws IOException* *public static void deleteFile(String targetDirectory) throws IOException* *public static void searchForFile(String targetDirectory)* *public static void buildFileList(String targetDirectory)*

Console Input and Output

Per the system requirements, the application is console based. Therefore, I used the `Scanner` class to

- Retrieve data from the console (using the *System.in* object to create a stream for theconsole input)
- Output messages to the console (using the *System.out* object and associated methods tooutput data to the console)

Given that there is so much interaction with the console, I chose to perform exception handling any timeI requested console input from the user. This application really taught me when to apply exception handling in a practical sense. Previously, I understood the concept at a high level, but didn't know when to apply it. Also, I took advantage of the clues provided by the Eclipse IDE.

Control Statements

The program utilizes the following control statements to direct the desired logic:

- *while* loop – Controls the program flow by prompting the User for main menu and the business options sub-menu, performing the desired operations, and terminates when theUser wishes to quite the program.
- *Switch* Statement – Executes the desired code statements associated with the main menuand the business level options sub-menu based on the value entered by the user.

File I/O

When the application launches, the first piece of user interaction is choosing the directory the application will use. The directory will be fixed and cannot be changed during the program. I created a "test mode" for the application, using the *D:\\Simplilearn\\LockedMe\\Test* directory on my laptop. However, the User can choose to use the *D:\\Simplilearn\\LockedMe\\Test* directory or provide a user-specified directory. Since the directory is console input, I implemented exception handling.

Populating the list of files in the directory

Once the target directory is set, a method called *buildFileList()* will populate the *fileList* collection for the first time. The *fileList* variable is a collection *(ArrayList*) of `File` objects.

```
29      private static List<File> fileList = new ArrayList<File>();
```

Since the *fileList* variable is a collection, I wanted to use the built-in *sort()* method provided bythe Java API for Collections. That way, I could call the *sort()* method on an as-needed basis.

Although I used a counter and *numFilesinDirectory* variables, I could have optimized my code by using the *size()* method provided by the *ArrayList*. Lastly, demonstrated understanding of *forEach* loopsand Collection iterators. I went back and forth about which packages to use for file I/O. The *java.nio.file*is recommended.However, the Java API also provides the older class in the *java.io.File* package, which also works. Iwanted to use a dedicated approach, selecting one approach for all the file I/O operations

Algorithms

This Application consist of five classes which are carefully designed there is less code and more code reusability.

- The *VirtualKey.java* class contains the driver code and the code where the user specific directory can be set.
- The Myscanner.java Class contains the *OpenScanner()* and *CloseScanner()* Method which consists of scanner objects they are written once and it is used through the entire program
- The Menu.java class contains the display screens and menu items that will be displayed on the console. Switching statements and Conditional statements of Java is used here.
- The MainMenuOperations.java class contains the methods which uses Java's conditional and switching statements. These code help the user to navigate between the Menus. The MainMenuOperations.java contains the *businessOptionsMenu*() method to navigate between the business level operations which includes adding a file, deleting a file, searching for a file and heading back to the main menu.
- The FileOperations.java class contains all the *retrieveFile*(), *addFile*(), *deleteFile*(), *searchForFile*(), *buildFileList*() which helps in retrieving the file, adding a file, deleting a file, building and sorting the files present in the directory respectively.

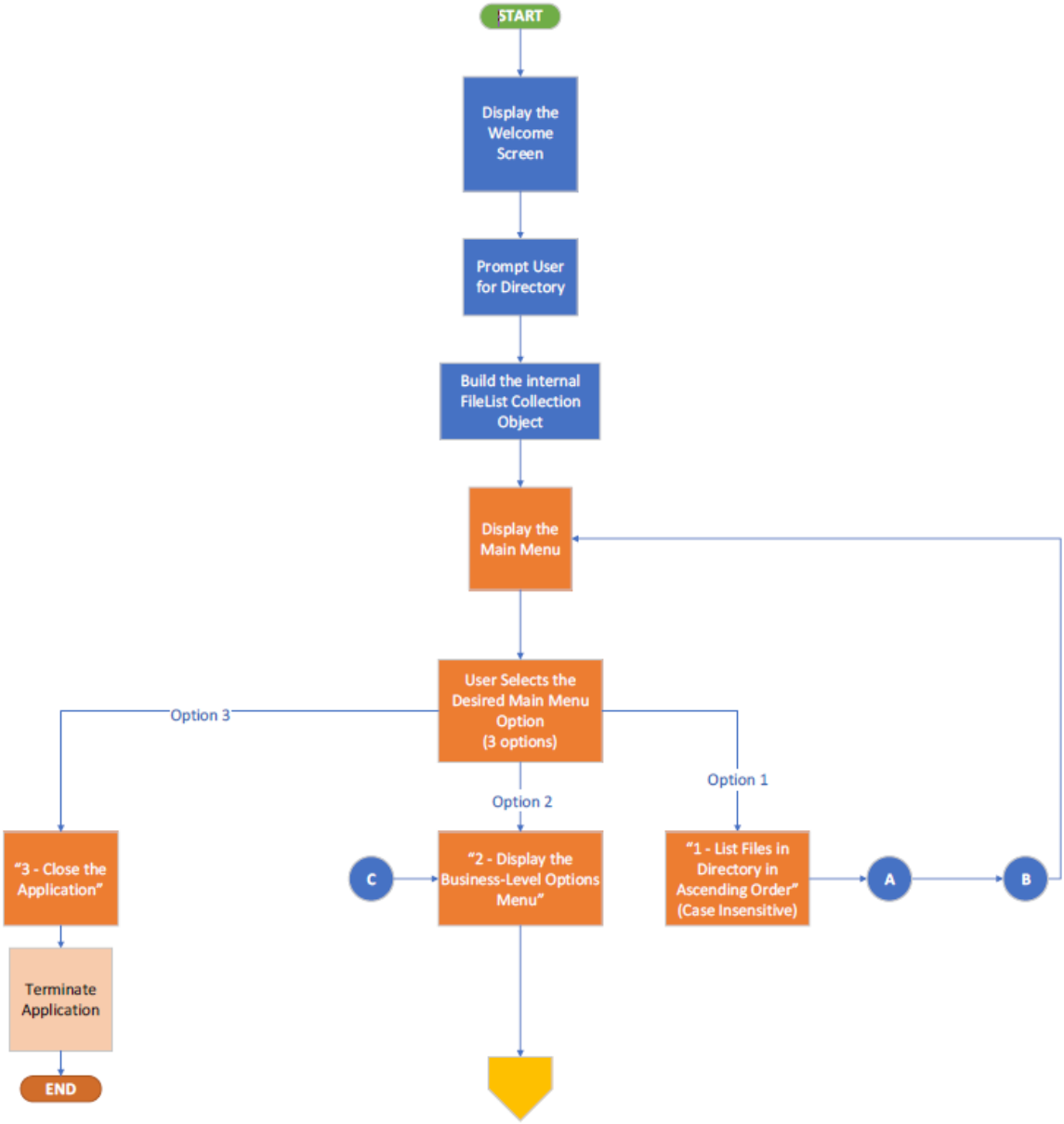
Data Structures Involved

Since this programs involves File operations, an *ArrayList* of Files which is a collection is used. The respective methods of the File Package is also used. And for sorting Java's inbuilt *sort*() from the collections is used.

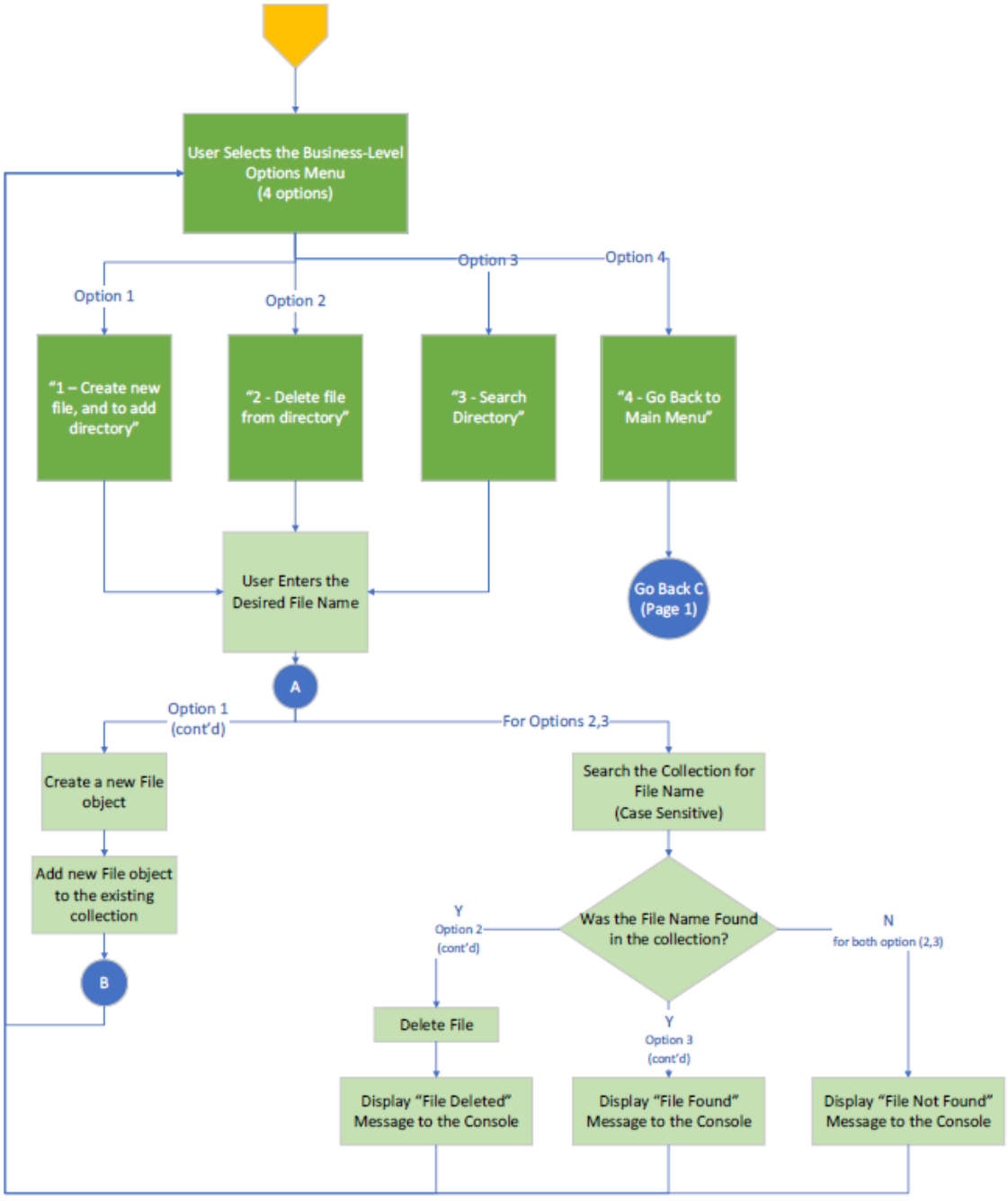
Flow Chart

Below Flow Chart explains the flow and logic of the application

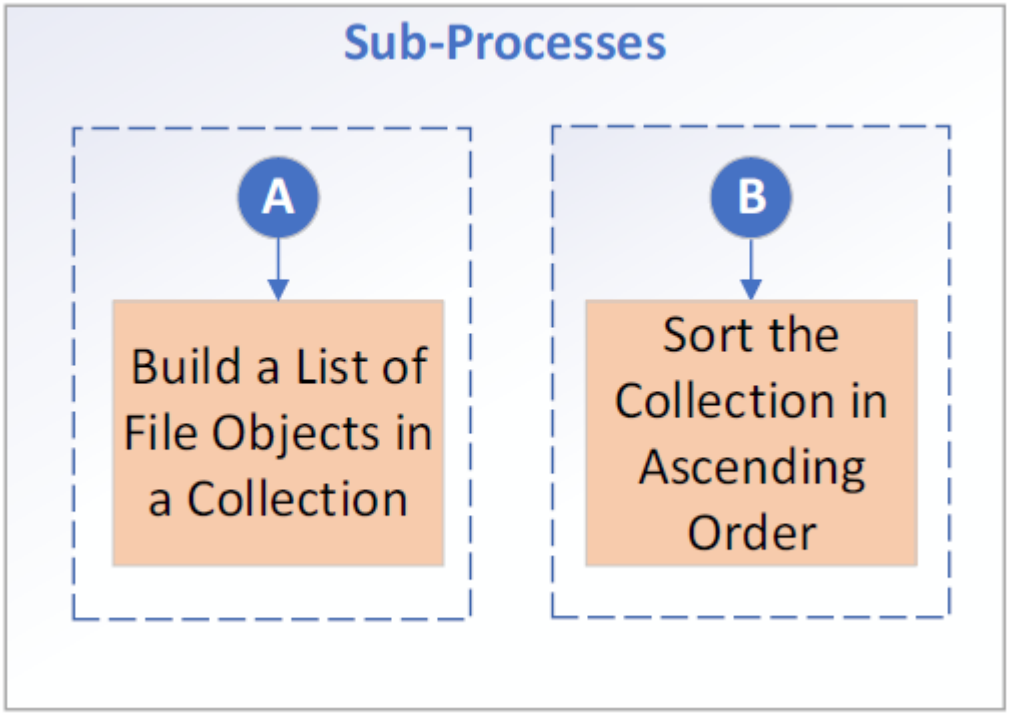
Application Flow Chart (page 1:3)



Application Flow Chart (page 2:3)



Application Flow Chart (page 3:3)



Future Improvement Areas

- In choosing the directory if a directory is not found the application is designed to terminate itself in future I am planning to prompt the user to try again
- In future additional features like creating a directory with directory and modifying the files which includes renaming the file and modifying the content in it are also in scope

GitHub Repository

I have pushed my code and associated documentation to the following GitHub repository:

https://github.com/Thiru97/Simplilearn2023.git

Code Screen Shots

VirtualKey.java

```java
package com.lockedme.virtualkey;
import java.io.File;
import java.util.InputMismatchException;

public class VirtualKey {

    // class instance variables
    public static String targetDirectory = null;

    // Parameterized and Default Constructors
    public static void setTargetDirectory(String path) {
        targetDirectory = path;
    }
    public static void setTargetDirectory() {
        targetDirectory = "D:\\Simplilearn\\LockedMe\\Test";
    }

    // Main program starts here and contains the driver code
    public static void main(String[] args) {
        MyScanner.openScanner();
        Menu.welcomeScreen();
        try {

            System.out.println(
                    "Do you like to run in Test directory(D:\\Test) or change to specified directory, Please enter 'yes' or 'no' ?");
            String test = MyScanner.sc.next();

            // Set the target directory
            if (test.equalsIgnoreCase("yes"))
                setTargetDirectory();
            else if (test.equalsIgnoreCase("no")) {
                System.out.println("Enter the target directory for LockedMe.com to use...");
                String path = MyScanner.sc.next();
                File f = new File(path);
```

```java
                // Checks if the specified path is available and is a directory
                if (f.exists() && f.isDirectory())
                    setTargetDirectory(path);
                else {
                    System.out.println("Invalid directory...");
                    System.out.println("Terminating Application");
                    System.exit(1);
                }
            } else {
                System.out.println("Entered Invalid option, Terminating Application...");
                System.exit(1);
            }

        } catch (InputMismatchException e) {
            MyScanner.sc.nextLine(); // clearing out the buffer
        } catch (Exception e) {
            e.printStackTrace();
        }
        // Displaying Main menu
        MainMenuOperations.mainMenu(targetDirectory);
        MyScanner.closeScanner();
S
    }// end main()

}// end class
```

MyScanner.java

```java
package com.lockedme.virtualkey;

import java.util.Scanner;

public class MyScanner {
    public static Scanner sc;

    public static void openScanner() {
        // Open a Scanner to read input from the console
        sc = new Scanner(System.in);
    }

    public static void closeScanner() {
        sc.close();
    }
}

```

Menu.java

```java
package com.lockedme.virtualkey;

import java.util.InputMismatchException;

public class Menu {

    public static void welcomeScreen() {

        System.out.println("\n*********************************************\n");
        System.out.println("Welcome to the Virtual Keys Repository of LockedMe.com");
        System.out.println("Version: 1.0 PROTOTYPE");
        System.out.println("Client: Lockers Pvt. Ltd");
        System.out.println("Full Stack Developer Name: Thirumavalaven");
        System.out.println("*********************************************\n");

    }// end welcome()

    public static int mainMenuOptions() {
        int r = 0; // added this to handle any exceptions

        String[] options = { "*********************************************",
                "*              MAIN MENU                    *", "*********************************************",
                "1. Display the current file names in ASCENDENING order", "2. Open Business Level Operations Menu",
                "3. Close the application", "*********************************************" };
```

```java
            // Display Main Menu Options
            for (int i = 0; i < options.length; i++) {
                System.out.println(options[i]);
            }

            try {
                System.out.println("Choose your option...");
                r = MyScanner.sc.nextInt();
                // couldn't return r from here; have to move it outside of the try-catch block
            } catch (InputMismatchException e) {
                MyScanner.sc.nextLine(); // consuming the input that was causing the exception, clearing the input
                                         // stream, and allowing the user to input something again
            } catch (Exception e) {
                e.printStackTrace();
            } // end catch
            return r;
        }// end mainMainOptions()
    }
```

MainMenuOperations.java

```java
package com.lockedme.virtualkey;

import java.util.InputMismatchException;

public class MainMenuOperations {

    public static void mainMenu(String targetDirectory) {
        int mainMenuChoice = 0;
        String appState = "c";

        System.out.println("You specified the following target directory: " + targetDirectory);
        // Prepare an ArrayList of the files available
        FileOperations.buildFileList(targetDirectory);
        // Main menu operations
        while (appState.equalsIgnoreCase("c")) {
            // Display the Main Menu Options
            try {
                mainMenuChoice = Menu.mainMenuOptions();
                System.out.println("Selected main menu option: " + mainMenuChoice);

                switch (mainMenuChoice) {
                case 1:
                    System.out.println("Retrieving files from " + targetDirectory + " in ascending order");
                    FileOperations.retrieveFiles(targetDirectory);
                    break;
                case 2:
                    FileOperations.businessOptionsMenu(targetDirectory);
                    break;
                case 3:
                    System.out.println("Closing application...");
                    System.exit(1);
                default:
                    System.out.println("Please enter a valid option..");
                    break;
                }// end switch
```

```java
            } catch (InputMismatchException e) {
                MyScanner.sc.nextLine();
            } // end catch
            catch (Exception e) {
                e.printStackTrace();
            } // end catch

            try {
                System.out.println("Enter 'c' to continue, 'x' to quit: ");
                appState = MyScanner.sc.next();
            } catch (InputMismatchException e) {
                System.out.println("you entered invalid input");
                MyScanner.sc.nextLine();
            } catch (Exception e) {
                e.printStackTrace();
            }

        } // end while

        if (appState.equalsIgnoreCase("x")) {
            System.out.println("Quitting the application...");
            System.exit(1);
        } else {
            System.out.println("Entered invalid sequence...Quitting the application...");
            System.exit(1);
        }

    }// end mainMenu()

}
```

FileOperations.java

```java
package com.lockedme.virtualkey;

import java.io.File;

public class FileOperations {

    public static Path myFile;
    public static List<File> fileList = new ArrayList<File>();
    public static int numFilesinDirectory = 0;
    public static String targetFileName = null;

    public static void businessOptionsMenu(String targetDirectory) {

        boolean loop = true;

        while (loop) {
            System.out.println("...............................................");
            System.out.println(".     Business Level Options Sub-Menu          .");
            System.out.println("...............................................");

            String[] options = { "1. Add new file", "2. Delete File (case sensisitive)",
                    "3. Search for File (case sensisitive)", "4. Go back to main menu",
                    "..............................................." };

            // Display the Business Level Options Menu to the Console
            for (int i = 0; i < options.length; i++) {
                System.out.println(options[i]);
            }

            try {
                // User will input their selection
                System.out.println("Choose your option...");
                int y = MyScanner.sc.nextInt();

                switch (y) {
                case 1:
                    System.out.println("Specify the file name to ADD to " + targetDirectory + ": ");
                    targetFileName = MyScanner.sc.next();
                    myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                    addFile(targetDirectory);
                    break;
                case 2:
                    System.out.println("Specify the file name to DELETE from " + targetDirectory + ": ");
                    targetFileName = MyScanner.sc.next();
                    myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                    deleteFile(targetDirectory);
                    break;
                case 3:
                    System.out.println("Specify the file name to SEARCH from " + targetDirectory + ": ");
                    targetFileName = MyScanner.sc.next();
                    myFile = Paths.get(targetDirectory + "\\" + targetFileName);
                    System.out.println("Searching for file: " + targetFileName);
                    searchForFile(targetDirectory);
                    break;
                case 4:
                    loop = false;
                    MainMenuOperations.mainMenu(targetDirectory);
                    break;
                default:
                    System.out.println("Please enter a valid option...");
                    break;
                }// end switch

            } catch (InputMismatchException e) {
                System.out.println("You entered invalid input! Try again..");
                MyScanner.sc.nextLine(); // consuming the input that was causing the exception, clearing the input
                                         // stream, and allowing the user to input something again
            } // end catch()
            catch (Exception e) {
                e.printStackTrace();
            } // end catch()
        } // end while

    }// end businessOptionsMenu()

```

Virtual Keys Repository Prototype Application

```
84●    public static void retrieveFiles(String targetDirectory) throws IOException {
85
86         int count = 0;
87         Path dirPath = Paths.get(targetDirectory);
88
89         if (Files.exists(dirPath) && Files.isDirectory(dirPath)) {
90             System.out.println("Directory: " + dirPath.toAbsolutePath());
91             System.out.println("Files: ");
92             DirectoryStream<Path> dirStream = Files.newDirectoryStream(dirPath);
93             for (Path p : dirStream) {
94                 if (Files.isRegularFile(p)) {
95                     System.out.println(p.getFileName()); // already sorted in ascending order
96                     count++;
97                 }
98             } // end for
99         } // end if
100
101        numFilesinDirectory = count;
102
103        System.out.println("There are " + numFilesinDirectory + " files in the directory.");
104
105    }// end retrieveFiles
106
```

```
107●    public static void addFile(String targetDirectory) throws IOException {
108         /*
109          * the requirement specifically stated "You can ignore the case sensitivity of
110          * the file names
111          */
112         Path filePath = Paths.get(targetDirectory, targetFileName);
113
114         if (Files.notExists(filePath)) {
115             Files.createFile(filePath);
116             System.out.println("File created successfully!"); // case sensitivity ignored
117
118             buildFileList(targetDirectory); // re-building the fileList after the addition
119             Collections.sort(fileList); // sort the resultant ArrayList in ascending order
120         } else {
121             System.out.println("File already exists");
122         }
123     }// end addFile()
124
```

```
125●    public static void deleteFile(String targetDirectory) throws IOException {
126
127         // The requirement was to implement case sensitive file delete functionality
128         File dirTest = new File(targetDirectory);
129
130         if (Files.exists(myFile)) {
131             System.out.println("Specified file exists...");
132             System.out.println("Do you really want to delete please confirm - y or n?");
133             String proceed = MyScanner.sc.next();
134
135             if (proceed.equalsIgnoreCase("y")) {
136                 for (File f : dirTest.listFiles()) {
137                     if (f.getCanonicalFile().getName().equals(myFile.getFileName().toString())) {
138                         Files.deleteIfExists(myFile);
139                         System.out.println("File deleted Successfully");
140                         buildFileList(targetDirectory); // re-building the fileList after the deletion
141                         Collections.sort(fileList); // sort the resultant ArrayList in ascending order
142                     } // end canonical check
143
144                 } // end for
145             } else {
146                 System.out.println("Not deleting the file per your request...");
147             }
148         } // end if(proceed)
149         else
150             System.out.println("File not found in " + dirTest + " ....");
151
152     }// end deleteFile()
```

```
153
154●    public static void searchForFile(String targetDirectory) {
155
156         File dirTest = new File(targetDirectory);
157         boolean fnf = true;
158
159         for (File f : dirTest.listFiles()) {
160             try {
161                 if (f.getCanonicalFile().getName().equals(myFile.getFileName().toString())) {
162                     System.out.println(myFile.getFileName().toString() + " exists in " + dirTest);
163                     fnf = false;
164                     break;
165                 }
166             } catch (IOException e) {
167                 e.printStackTrace();
168             }
169         } // end for()
170
171         if (fnf != false)
172             System.out.println(myFile.getFileName().toString() + " DOES NOT exist in " + dirTest + "...");
173
174     }// searchForFile()
175
```

```
175
176●    public static void buildFileList(String targetDirectory) {
177         int count = 0;
178
179         File dirTest = new File(targetDirectory);
180
181         for (File file : dirTest.listFiles()) {
182             fileList.add(file);
183             count++;
184         }
185
186         numFilesinDirectory = count;
187
188     }// end buildFileList()
189
190 }
```

Output Example

Test Case Scenario: Launch Program – Use Test Mode

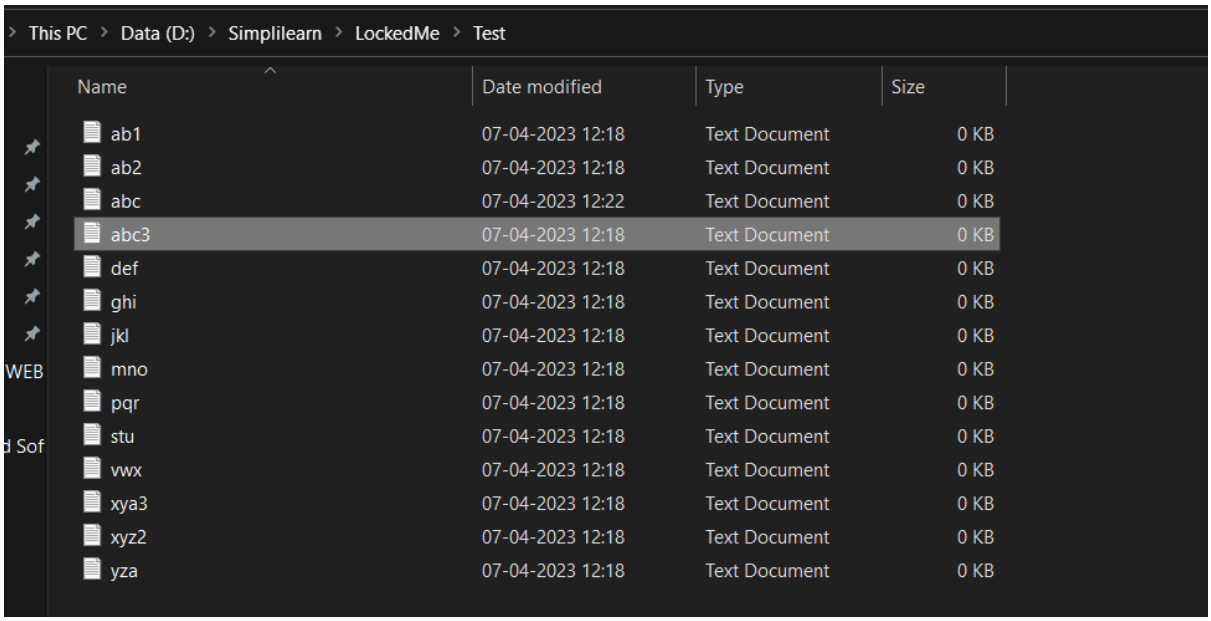
Using the default test Directory as the main Directory and entering the main menu correctly

```
Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Thirumavalaven
*********************************************

Do you like to run in Test directory(D:\Test) or change to specified directory, Please enter 'yes' or 'no' ?
yes
You specified the following target directory: D:\Simplilearn\LockedMe\Test
*********************************************
*               MAIN MENU                   *
*********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
*********************************************
Choose your option...
```

Choosing Option 1 to list Files in the directory in ascending order

TEST Directory Folder structure in Personal computer



OUTPUT

```
*********************************************
*              MAIN MENU                    *
*********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
*********************************************
Choose your option...
1
Selected main menu option: 1
Retrieving files from D:\Simplilearn\LockedMe\Test in ascending order
Directory: D:\Simplilearn\LockedMe\Test
Files:
ab1.txt
ab2.txt
abc.txt
abc3.txt
def.txt
ghi.txt
jkl.txt
mno.txt
pqr.txt
stu.txt
vwx.txt
xya3.txt
xyz2.txt
yza.txt
There are 14 files in the directory.
Enter 'c' to continue, 'x' to quit:
```

Entering “c” to continue and entering option “2” to views Business level options sub-menu

```
Enter 'c' to continue, 'x' to quit:
c
*********************************************
*               MAIN MENU                   *
*********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
*********************************************
Choose your option...
2
Selected main menu option: 2
.............................................
.     Business Level Options Sub-Menu       .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
```

Choosing option “1” and Adding a file named “hello.txt”

```
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
1
Specify the file name to ADD to D:\Simplilearn\LockedMe\Test:
hello.txt
File created successfully!
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
```

Choosing option “3” and Searching for a file “hello.txt”

```
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
3
Specify the file name to SEARCH from D:\Simplilearn\LockedMe\Test:
hello.txt
Searching for file: hello.txt
hello.txt exists in D:\Simplilearn\LockedMe\Test
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
```

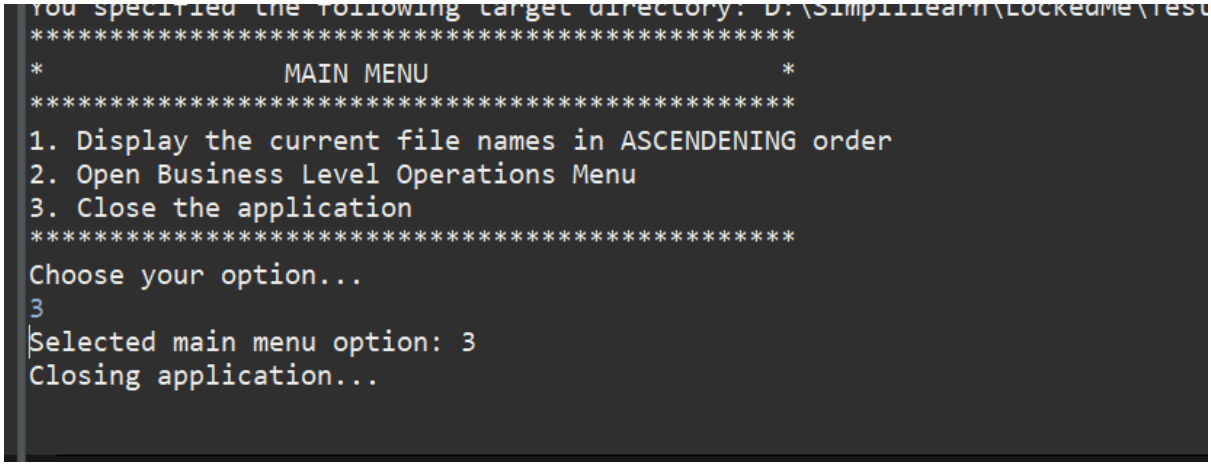
Choosing option “2” and deleting a file “hello.txt”

```
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
2
Specify the file name to DELETE from D:\Simplilearn\LockedMe\Test:
hello.txt
Specified file exists...
Do you really want to delete please confirm - y or n?
y
File deleted Successfully
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
```

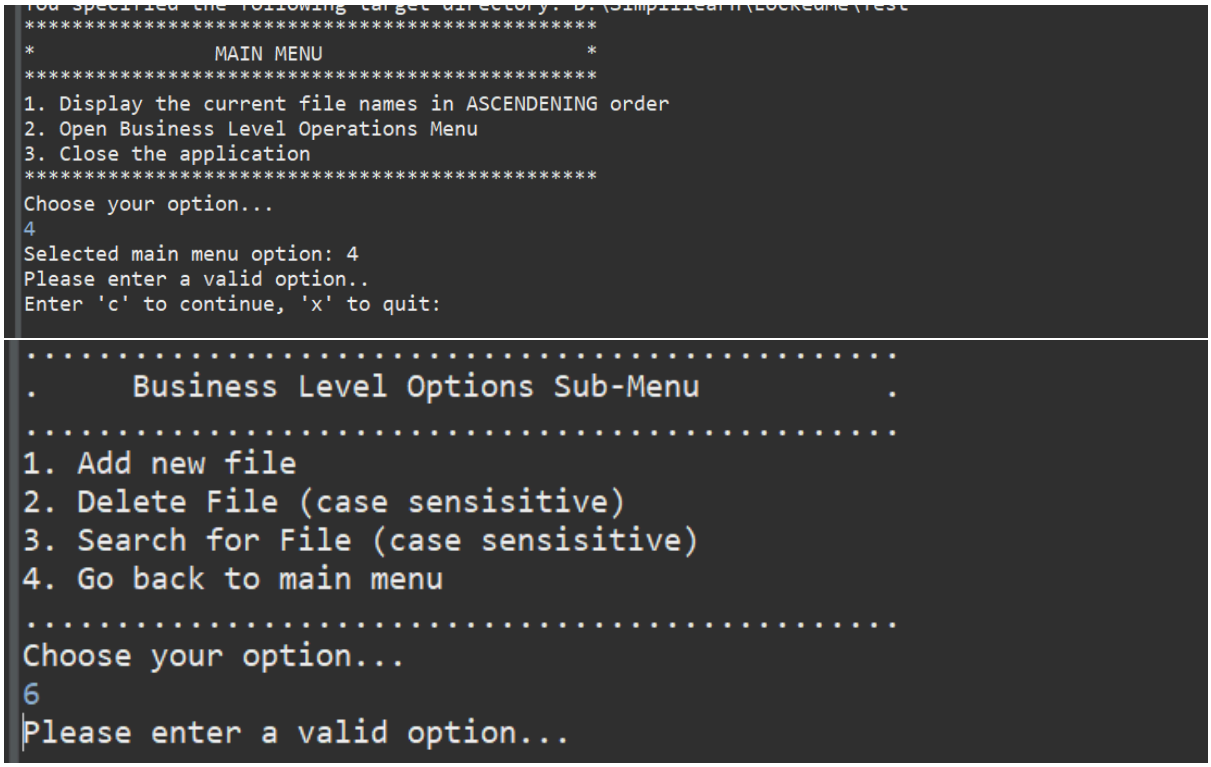
Choosing option “4” and heading back to main menu

```
.............................................
.     Business Level Options Sub-Menu         .
.............................................
1. Add new file
2. Delete File (case sensisitive)
3. Search for File (case sensisitive)
4. Go back to main menu
.............................................
Choose your option...
4
You specified the following target directory: D:\Simplilearn\LockedMe\Test
*********************************************
*              MAIN MENU                      *
*********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
*********************************************
Choose your option...
```

Choosing option “3” and closing the application



Giving Options which are not in the menu and checking the exception handling



Giving a different directory (D:\\Test2) instead of test directory and listing its files in ascending order

```
*********************************************

Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Thirumavalaven
*********************************************

Do you like to run in Test directory(D:\Test) or change to specified directory, Please enter 'yes' or 'no' ?
no
Enter the target directory for LockedMe.com to use...
D:\\Test2
You specified the following target directory: D:\\Test2
*********************************************
*               MAIN MENU                     *
*********************************************
1. Display the current file names in ASCENDENING order
2. Open Business Level Operations Menu
3. Close the application
*********************************************
Choose your option...
1
Selected main menu option: 1
Retrieving files from D:\\Test2 in ascending order
Directory: D:\Test2
Files:
abc.txt
art.txt
hello.txt
zeppo.txt
There are 4 files in the directory.
Enter 'c' to continue, 'x' to quit:
```

Giving a different directory which doesn’t exist

```
*********************************************

Welcome to the Virtual Keys Repository of LockedMe.com
Version: 1.0 PROTOTYPE
Client: Lockers Pvt. Ltd
Full Stack Developer Name: Thirumavalaven
*********************************************

Do you like to run in Test directory(D:\Test) or change to specified directory, Please enter 'yes' or 'no' ?
no
Enter the target directory for LockedMe.com to use...
D:\\Test3
Invalid directory...
Terminating Application
```