Project Specification and Sprint Work – LockeMe.com (Phase I project)

Revision History			
Version no	Description	Author	Date of change
v0	Initial version of LockMe.com project	Nandakumar R	10 Aug 2021

Table of Contents

Introduction:

Company Lockers Pvt. Ltd. need to digitize their products and chose LockedMe.com as their first project to start with. Need to develop a prototype of the application. The prototype of the application needs to be presented to the relevant stakeholders for budget approval.

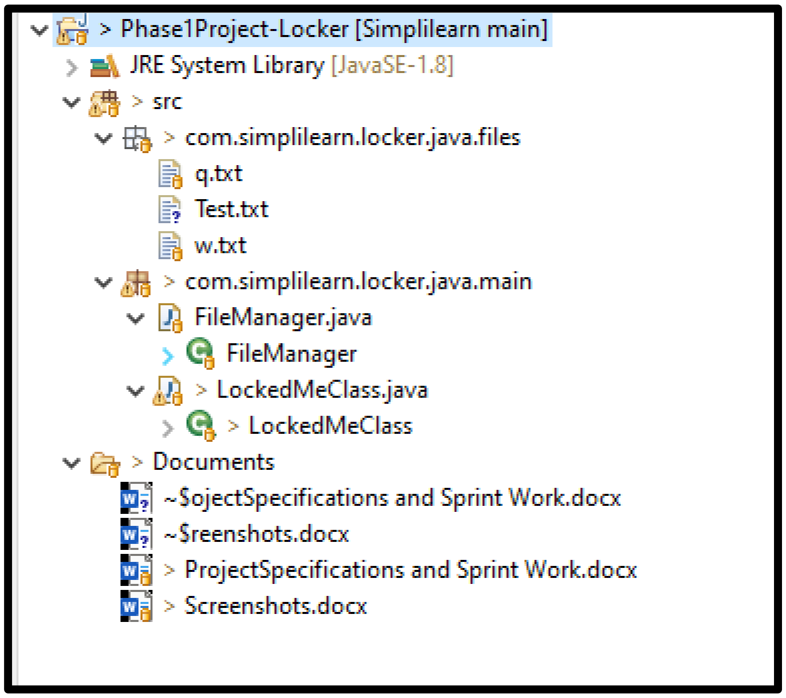
Product's capabilities:

LockMe.com product need to have below specifications as follows

1. Retrieving the file names in an ascending order
2. 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
3. Option to close the application.

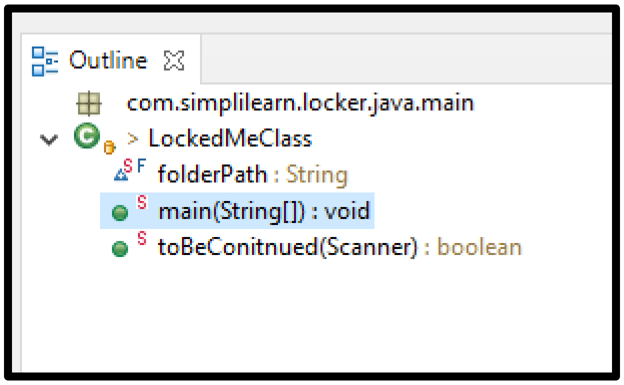
Appearance:

Project Folder Structure:



```
> Phase1Project-Locker [Simplilearn main]
  JRE System Library [JavaSE-1.8]
  > src
    > com.simplilearn.locker.java.files
        q.txt
        Test.txt
        w.txt
    > com.simplilearn.locker.java.main
      FileManager.java
        FileManager
      > LockedMeClass.java
        > LockedMeClass
  > Documents
      ~$ojectSpecifications and Sprint Work.docx
      ~$reenshots.docx
      > ProjectSpecifications and Sprint Work.docx
      > Screenshots.docx
```

Class Outline:


Outline

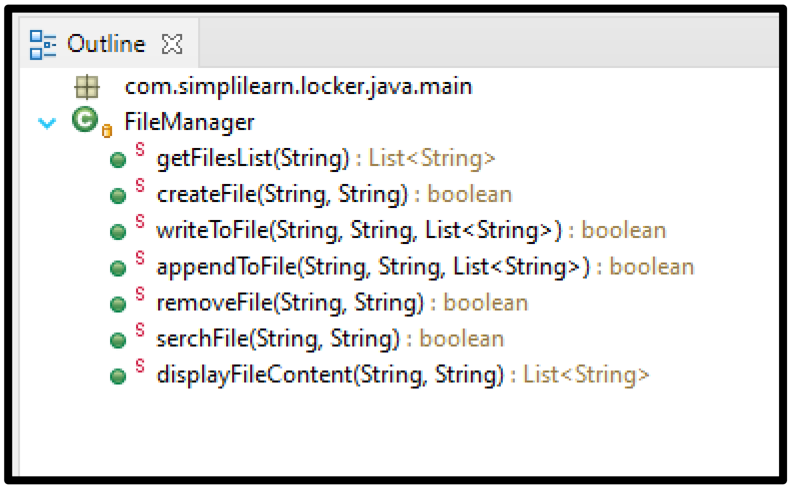
com.simplilearn.locker.java.main

> LockedMeClass

folderPath : String

main(String[]) : void

toBeConitnued(Scanner) : boolean


Outline

com.simplilearn.locker.java.main

FileManager

getFilesList(String) : List<String>

createFile(String, String) : boolean

writeToFile(String, String, List<String>) : boolean

appendToFile(String, String, List<String>) : boolean

removeFile(String, String) : boolean

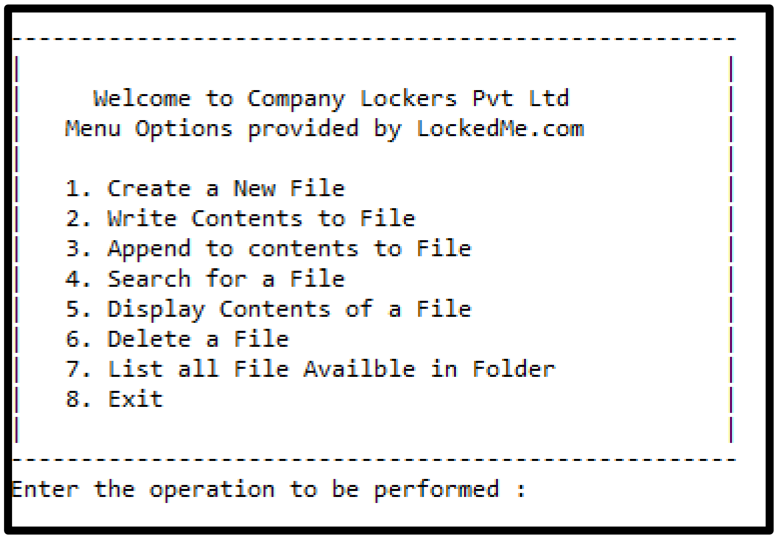
serchFile(String, String) : boolean

displayFileContent(String, String) : List<String>

Menu Options:

Below are the menu options provided to the application users to perform

```
1. Create a New File
2. Write Contents to File
3. Append to contents to File
4. Search for a File
5. Display Contents of a File
6. Delete a File
7. List all File Availble in Folder
8. Exit
```



```
----------------------------------------------------
|                                                  |
|     Welcome to Company Lockers Pvt Ltd           |
|   Menu Options provided by LockedMe.com          |
|                                                  |
|   1. Create a New File                           |
|   2. Write Contents to File                      |
|   3. Append to contents to File                  |
|   4. Search for a File                           |
|   5. Display Contents of a File                  |
|   6. Delete a File                               |
|   7. List all File Availble in Folder            |
|   8. Exit                                        |
|                                                  |
----------------------------------------------------
Enter the operation to be performed :
```

User interactions:

Users can interact to the application via console to provide options to perform any operations provided in the application.

Number and duration of sprints:

Sprints planned for this project is as in below Table:

Sprint No	Module Planned	Start Date	End Date	Hours Planned
1	Prototype Designing with Menu Options	08 Aug 2021	08 Aug 2021	2
2	Working on Menu options	09 Aug 2021	10 Aug 2021	3
3	Final Demo to Customer for Business sign-off	11 Aug 2021	11 Aug 2021	1

Git and GitHub account:

Git account where the source code is maintained: https://github.com/NandakumarRangnathan/Phase-1---Java-Class/tree/main/Phase1Project-Locker

Java concepts:

Java concepts being used in the project as below

1. Collections
2. List
3. Arrays
4. File and File Handling techniques
5. Scanner
6. Exception Handling

Source Code:

Below is the source code form the project

LockedMeClass.class

```
package com.simplilearn.locker.java.main;


import java.io.IOException;

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.List;

import java.util.Scanner;


public class LockedMeClass {


	static final String folderPath = "C:\\Needs\\Simplilearn\\Phase1Project-
Locker\\src\\com\\simplilearn\\locker\\java\\files";


	public static void main(String[] args) {


		Scanner obj = new Scanner(System.in);

		String fileName;

		int lineCount, performOperation;

		boolean isDeleted = false, isCreated = false, isAvailable = false, needToContinue
= false, isWritten = false;

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


		String welcomeMessage = "    " + "\n---------------------------------------------------"

				+ "\n|                                             |"

				+ "\n|     Welcome to Company Lockers Pvt Ltd          |"

				+ "\n|   Menu Options provided by LockedMe.com         |"
```

```
                                        + "\n|                                    |"
                                        + "\n|   1. Create a New File             |"
                                        + "\n|   2. Write Contents to File        |"
                                        + "\n|   3. Append to contents to File    |"
                                        + "\n|   4. Search for a File             |"
                                        + "\n|   5. Display Contents of a File    |"
                                        + "\n|   6. Delete a File                 |"
                                        + "\n|   7. List all File Availble in Folder   |"
                                        + "\n|   8. Exit                          |"
                                        + "\n|                                    |"
                                        + "\n--------------------------------------------------";

                    do {

                              try {
                                        System.out.println(welcomeMessage);

                                        System.out.println("Enter the operation to be performed : ");
                                        performOperation = obj.nextInt();

                                        switch (performOperation) {

                                        case 1:
                                                  // Creating a New file in the location.
                                                  System.out.println("Enter File Name to be created : ");
                                                  fileName = obj.next();

                                                  isCreated = FileManager.createFile(fileName,
folderPath);
```

```
                    if (isCreated)
                        System.out.println("File created with name : " +
fileName);

                    needToContinue = toBeConitnued(obj);

                    break;

                case 2:
                    // Writting contents to the file which user secified
                    System.out.println("Enter the file Name to write the
contents to it : ");
                    fileName = obj.next();

                    System.out.println("Enter the number of lines to be
written to File : ");
                    lineCount = obj.nextInt();

                    System.out.println("Enter the content : ");
                    for (int i = 0; i <= lineCount; i++) {
                        content.add(obj.nextLine());
                    }

                    isWritten = FileManager.writeToFile(folderPath,
fileName, content);

                    if (isWritten)
                        System.out.println("Written the contents the
File");
                    else
                        System.out.println("Not written to the file");
```

```
				needToContinue = toBeConitnued(obj);
				break;

			case 3:
				System.out.println("Enter the file Name to append the
contents to it : ");
				fileName = obj.next();

				System.out.println("Enter the number of lines to be
appended to File : ");
				lineCount = Integer.parseInt(obj.next());

				System.out.println("Enter the content of line : ");
				for (int i = 0; i <= lineCount; i++) {
					content.add(obj.nextLine());
				}

				if (FileManager.serchFile(folderPath, fileName))
					isWritten =
FileManager.appendToFile(folderPath, fileName, content);

				if (isWritten)
					System.out.println("Written the contents the
File");
				else
					System.out.println("We dont find the file to
append the content");

				needToContinue = toBeConitnued(obj);
				break;
			case 4:
				// Searching a File
```

```
                    System.out.println("Enter the file name to be searched : ");
                    fileName = obj.next();

                    isAvailable = FileManager.serchFile(folderPath, fileName);
                    if (isAvailable)
                        System.out.println("File is present in the folder");
                    else
                        System.out.println("File not present in the folder");

                    needToContinue = toBeConitnued(obj);
                    break;

                case 5:
                    // Displaying content a File
                    System.out.println("Enter the file name to be displayed : ");
                    fileName = obj.next();

                    isAvailable = FileManager.serchFile(folderPath, fileName);
                    if (isAvailable) {

                        List<String> diplayList = FileManager.displayFileContent(folderPath, fileName);

                        if (diplayList.isEmpty()) {
                            System.out.println("File has no content to disply Now");
                        } else {
```

```
                                                            diplayList.forEach((n) ->
System.out.println(n));
                                                    }
                                            } else
                                                    System.out.println("File not present in the
folder");

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

                                            needToContinue = toBeConitnued(obj);
                                            break;

                                    case 6:
                                            // File to be deleted.
                                            System.out.println("Enter the file name to be deleted :
");
                                            fileName = obj.nextLine();

                                            isDeleted = FileManager.removeFile(folderPath,
fileName);
                                            if (isDeleted)
                                                    System.out.println(fileName + " - File deleted
successfully");
                                            else
                                                    System.out.println("We dont see file named :" +
fileName);

                                            needToContinue = toBeConitnued(obj);
                                            break;

                                    case 7:
                                            // Getting List of Files in the Folder in ascending order.
```

```
					List<String> fileList =
FileManager.getFilesList(folderPath);

					if (!fileList.isEmpty()) {
						System.out.println("Files available in the folder
are below : ");
						fileList.forEach((n) -> System.out.println(n));
					} else {
						System.out.println("Folder is empty.");
					}
					needToContinue = toBeConitnued(obj);
					break;

				case 8:
					System.exit(0);
					needToContinue = false;
					break;

				default:
					System.out.println("You have provided a wrong
option.");
					needToContinue = toBeConitnued(obj);
					break;
				}
			} catch (InputMismatchException e) {
				System.out.println("Please input a valid Menu option to
perform");
				main(args);
			}

		} while (needToContinue);
```

```
	}

	public static boolean toBeConitnued(Scanner obj) {

		System.out.println("Do you want to Continue : Yes or No");
		String userInput = obj.next();
		if (userInput.equalsIgnoreCase("Yes"))
			return true;
		else
			return false;
	}

}
```

FileManager.class:

```
package com.simplilearn.locker.java.main;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class FileManager {
	public static List<String> getFilesList(String filePath) {

		// Variable declaration
		List<String> al = new ArrayList<String>();

		// Creating File Object
		File fileList = new File(filePath);

		// Getting list of files available in folder
		String[] listOfFiles = fileList.list();

		// looping through file array
		for (String file : listOfFiles)
			al.add(file);

		Collections.sort(al);

		return al;
	}

	public static boolean createFile(String fileName, String filePath) {

		File file = new File(filePath, fileName);
		boolean isFielCreated = false;

		try {
			if (file.createNewFile())
				isFielCreated = true;

		} catch (IOException e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                return isFielCreated;
        }

        public static boolean writeToFile(String path, String fileName, List<String> fileContent)
{

                try {
                        File f = new File(path, fileName);
                        FileWriter fw = new FileWriter(f);

                        if (!f.exists())
                                f.createNewFile();

                        for (String s : fileContent) {
                                if (s.length() > 0)
                                        fw.write(s + "\n");
                        }

                        fw.close();
                        return true;

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return false;
                }
```

```
	}

	public static boolean appendToFile(String path, String fileName, List<String>
fileContent) {

		try {
			BufferedWriter out = new BufferedWriter(new FileWriter(path + "\\" +
fileName, true));
			fileContent.forEach((n) -> {
				try {
					if (n.length() > 0)
						out.write(n + "\n");
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			});
			// out.write(str);
			out.close();
			return true;
		} catch (Exception e) {
			// TODO: handle exception
		}

		return false;
	}

	public static boolean removeFile(String filePath, String fileName) {

		File fl = new File(filePath + "\\" + fileName);
```

```
                try {

                        if (fl.delete())
                                return true;

                } catch (Exception e) {
                        // TODO: handle exception
                }

                return false;
        }

        public static boolean serchFile(String filePath, String fileName) {

                File fl = new File(filePath + "//" + fileName);

                try {

                        if (fl.exists())
                                return true;

                } catch (Exception e) {
                        // TODO: handle exception
                }
                return false;
        }

        public static List<String> displayFileContent(String filePath, String fileName){

```

```
		BufferedReader br;
		List<String> al = new ArrayList<String>();
		try {
			br = new BufferedReader(new FileReader(filePath + "//" + fileName));
			String line;
				while ((line = br.readLine()) != null) {
					al.add(line);
				}


		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		return al;
	}

}
```