LockedMe.com

August 14, 2021

Durga Pathak
Simplilearn.com
Sr. Java Developer

LockedMe.com

(Sprint work and Project Specification)

Version History

Author	Durga Pathak
Purpose	Application screen shots
Date	August 14, 2021
Version	1.0

Table of Contents

1. Modules in the project 3
2. Sprint Work 3
3. Project GitHub link 3
4. Core Concept Used 4
5. Folder Structure 5
6. LockedMeProject.java 5
7. FileManager.java 10

1. Modules in the project

1. **Display All files**
2. **Add File**
3. **Delete File**
4. **Search File**

2. Sprint Work

Sprint Number	**Modules**
1	**Display All Files** (Displays the names of all the files in a given folder in alphabetical order) **Add File** (Creates a files in a folder and adds contents to the file) **Delete File** (Deletes the file from a given folder)
2	**Search File** (Searches a file in a folder. Informs user whether files exists or does not exist in a folder) **Testing** (Verify all modules are working as expected) **Documentation** (Create project specifications with screen shots of each modules)

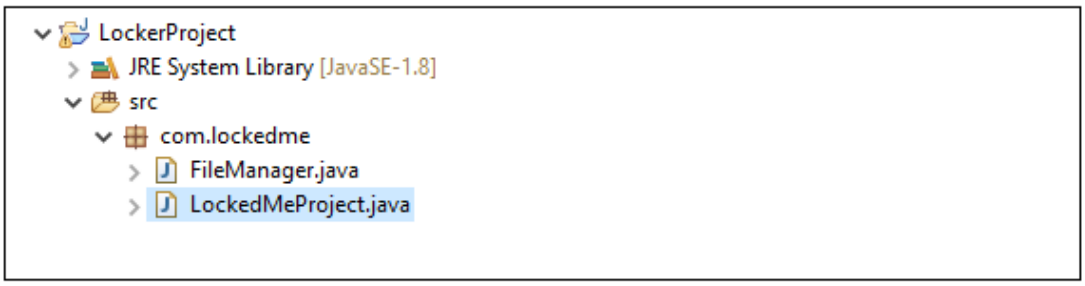
3. Project GitHub link

Repository Name	Project-1
Repository URL	https://github.com/durgapathak18/Project-1

4. Core Concept Used

- Object Oriented Programming
- File Handling
- Exception Handling
- Java Loops
- Java 8 stream
- Arrays and Collections
- Scanner

5. Folder Structure



6. LockedMeProject.java

```
package com.lockedme;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Driver class
 *
 * @author Durga Pathak
 *
 */
public class LockedMeProject
{

	static final String FOLDER_LOCATION = "C:\\Users\\15124\\eclipse-workspace\\LockedMeFiles";

	public static void main(String[] args)
	{
		// Variable declaration
		int proceed = 1;
		int choice;
```

```
        String userInput;

        // Create scanner object to get input from user
        Scanner scanner = new Scanner(System.in);

        do
        {
                // Display Menu
                displayMenu();

                System.out.println("Enter your choice:");
                userInput = scanner.nextLine();

                // Logic to let user know if they key invalid entry.
                while (userInput == null || userInput.length() == 0 || !userInput.matches("\\d"))
                {
                        System.err.println("Invalid choice. \nEnter number between 1 and 5.");
                        displayMenu();
                        System.out.println("Enter your choice:");
                        userInput = scanner.nextLine();
                }

                choice = Integer.parseInt(userInput);

                switch (choice) {
                case 1: getAllFiles();
                                break;
                case 2: createFiles(scanner);
                                break;
                case 3: deleteFile(scanner);
                                break;
                case 4: searchFile(scanner);
                                break;
                case 5:
                                System.out.println("\nThank you for visiting lockedme.com \nVisit us again.");
                                System.exit(0);
                                break;
                default:
                                System.err.println("\nLooks like you picked invalid option. Try again.");
                                break;
                }
        } while (proceed > 0);

        scanner.close();
}

private static void displayMenu()
```

```
{
        System.out.println("\n\n*************************************************************");
        System.out.println("\t\t\tLockedMe.com");
        System.out.println("*************************************************************");
        System.out.println("1. Display all files");
        System.out.println("2. Add new file");
        System.out.println("3. Delete a file");
        System.out.println("4. Search a file");
        System.out.println("5. Exit");
        System.out.println("*************************************************************");

}

/**
 * Gets the name of all files.
 */
private static void getAllFiles()
{
        // Gets the name of all files in the given folder
        List<String> namesOfAllFiles = FileManager.getNamesOfAllFiles(FOLDER_LOCATION);

        // Print file names
        namesOfAllFiles.stream().forEach(name -> System.out.println(name));
}

/**
 * This method creates a file
 */
private static void createFiles(Scanner scanner)
{
        // Variable declarations
        String fileName;
        int totalNumOfLines;
        boolean isContentSaved = false;
        String userInput;
        List<String> listOfContent = new ArrayList<String>();

        // Read file name from user
        System.out.println("Enter File Name: ");
        fileName = scanner.nextLine();

        // Ask user for how many lines of content they intend to write to a file
        System.out.print("Enter the number of lines in the file: ");
        userInput = scanner.nextLine();

        // If user enters invalid input, don't save content
        if (userInput == null || userInput.length() == 0 || !userInput.matches("\\d"))
```

```
		{
			isContentSaved = false;
		}
		else
		{
			totalNumOfLines = Integer.parseInt(scanner.nextLine());

			// Collect content for each line
			for (int i = 1; i <= totalNumOfLines; i++)
			{
				System.out.println("Enter content for line " + i);
				listOfContent.add(scanner.nextLine());

			}

			// Save content to the file
			isContentSaved = FileManager.writeContentToFile(FOLDER_LOCATION, fileName,
listOfContent);
		}

		if (isContentSaved)
		{
			System.out.println("\nContent successfully saved to " + fileName);
		} else
		{
			System.err.println("Unable to write content to file. \nCheck input and try again. If issue
persists, contact support@simplilearn.com");
		}

	}

	/**
	 * Deletes a file in the folder
	 */
	private static void deleteFile(Scanner scanner)
	{
		// Variable declaration
		String fileName;
		System.out.println("Enter file name to delete: ");
		fileName = scanner.nextLine();

		boolean isFileDeleted = FileManager.deleteFile(FOLDER_LOCATION, fileName);

		if (isFileDeleted)
		{
			System.out.println("\n" + fileName + " is deleted successfully.");
		}
```

```
            else
            {
                System.err.println("\nUnable to delete specified file. Verify if given file exist and try again.");
            }

        }

        /**
         * Searches a file in the folder
         */
        private static void searchFile(Scanner scanner)
        {

            // Variable declaration
            String fileName;
            System.out.println("Enter file name to search: ");
            fileName = scanner.nextLine();

            boolean isFileDeleted = FileManager.searchFile(FOLDER_LOCATION, fileName);

            if (isFileDeleted)
            {
                System.out.println("\n" + fileName + " exists in the folder");
            }
            else
            {
                System.err.println("\n"  + fileName + " does not exist in specified folder.");
            }

        }

}
```

7. FileManager.java

```
package com.lockedme;

import java.io.File;
import java.io.FileWriter;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

/**
 * This class contains methods to manage files
 *
 * @author Durga Pathak
 *
 */
public class FileManager
{
	/**
	 * This method returns the list of names of files in given folder.
	 *
	 * @param folderLocation
	 * @return List of File Names
	 */
	public static List<String> getNamesOfAllFiles(final String folderLocation)
	{
		// Create File Object
		File file = new File(folderLocation);

		// Gets all files into an array
		File[] listOfFiles = file.listFiles();

		// Collect the name of each file in the given folder location
		List<String> fileNames = Arrays.asList(listOfFiles).stream().map(lof ->
lof.getName()).collect(Collectors.toList());

		// Sort files in alphabetical order
		fileNames.stream().sorted();
		return fileNames;

	}

	/**
	 * This method writes the contents to a given file name in specified folder location.
	 *
```

```
	 * @param folderLocation
	 * @param fileName
	 * @param contentList
	 * @return boolean (if file saved successfully or not)
	 */
	public static boolean writeContentToFile(final String folderLocation, final String fileName, final List<String>
contentList) {

		try
		{
			// Create File Object
			File file = new File(folderLocation, fileName);
			FileWriter fileWriter = new FileWriter(file);

			// Loop through each file content and write it to the file
			for (String content : contentList) {
				fileWriter.write(content + "\n");

			};

			// Close file writer
			fileWriter.close();
			return true;

		} catch (Exception e)
		{
			return false;
		}

	}

	/**
	 * This method deletes the given file from the given folder.
	 *
	 * @param folderLocation
	 * @param fileName
	 * @return boolean
	 */
	public static boolean deleteFile(final String folderLocation, final String fileName)
	{
		// File to delete
		File file = new File(folderLocation + "\\" + fileName);

		try
		{
			// Delete file
			if (file.delete())
```

```
                {
                        return true;
                }
        } catch (Exception e)
        {
                return false;
        }
        return false;

    }

    /**
     * This method searches files in a given folder and informs the user if file exists or not.
     *
     * @param folderLocation
     * @param fileName
     * @return boolean
     */
    public static boolean searchFile(final String folderLocation, final String fileName) {

        // Create file object with given folder location and file name
        File file = new File(folderLocation + "\\" + fileName);

        // Return if file exists or not
        return file.exists();

    }

}
```