Company Lockers Pvt. Ltd.

Product LockedMe.com

Prototype of the Application

Name : Anurag Sharma

GitHub : https://github.com/Instantgaming2356/JAVAFSD-Project01

The prototype of the application is operated as a CLI (Command Line Program) without GUI. Its usage is to do file operations such as create new files along with content, delete a file or search a file from a specified directory and list them afterward in sorting order.

The implementation is done with the help of Java 8 and IDE IntelliJ.

Sprint Planning

The Implementation is done in two sprints which are mentioned below:

Sprint 1:

- Clarify the specification and requirements.
- Implement view content mechanism.
- Implement list of all files in sorted order.
- Implement functionality to close the program safely.

Sprint 2:

- Implement functionality to add create files along the content.
- Implement functionality to delete a file if it is present in that user specified directory.
- Implement functionality to search a file in the same directory.
- Documentation

Documentation of the functionality:

Here is the different Options that user can choose to perform certain file operation of the program.

Welcome Screen

This is the first screen that user will interact.

```
***LOCKEDME.COM***
->Developed By Anurag Sharma
============================

Directory : src/Storage/

Main Menu
1. Show Files
2. Show File Options Menu
3. Quit
Enter the choice :
```

List all files

This option will let user to see list of files in the specified directory in sorted order.

```
Main Menu
1. Show Files
2. Show File Options Menu
3. Quit
Enter the choice : 1
B.txt
Demo.txt
Sample.txt
```

File Operations

This option will let user to provide several file operations with.

```
Main Menu
1. Show Files
2. Show File Options Menu
3. Quit
Enter the choice : 2

File Options Menu
1. Add A File
2. Delete A File
3. Search A File
4. Return to Menu
Enter the choice :
```

Create a file

This will allow user to create a file along with content inside it.

```
File Options Menu
1. Add A File
2. Delete A File
3. Search A File
4. Return to Menu
Enter the choice : 1
Adding File...
Please Enter the file name with extension to Create : Test1.txt
File Created!!!
Do You Want to add Content
Y
Enter the New Content : Hi Simplilearn
Data is successfully added to the file.
```

Delete a file

This will allow user to delete a file if it is present otherwise it will send a appropriate message.

```
File Options Menu
1. Add A File
2. Delete A File
3. Search A File
4. Return to Menu
Enter the choice : 2
Deleting File...
Please Enter the file name with extension to Delete : B.txt
File Deleted!!!
```

Search a file

This will allow user to input a file name along with extension to begin the search procedure and give back the appropriate result.

```
File Options Menu
1. Add A File
2. Delete A File
3. Search A File
4. Return to Menu
Enter the choice : 3
Searching the file...
Please Enter the Filename:
Test1.txt
You are searching for a file named: Test1.txt
Found Test1.txt
```

Quit

This will allow user to exit from the program safely.

```
File Options Menu
1. Add A File
2. Delete A File
3. Search A File
4. Return to Menu
Enter the choice : 4

Main Menu
1. Show Files
2. Show File Options Menu
3. Quit
Enter the choice : 3

Thank You!!!

Process finished with exit code 0
```

Source Code

1: Welcome Component

```java
package Welcome;

import Services.DirectoryService;
import Services.ScreenService;
import Shared.Screen;

import java.util.ArrayList;

3 usages  Anurag Sharma (anushar4) *
public class Welcome implements Screen {

    1 usage
    final String welcomeText = "***LOCKEDME.COM***";
    1 usage
    final String developerText = "->Developed By Anurag Sharma";

    4 usages
    private final ArrayList<String> options = new ArrayList<>();


    3 usages  Anurag Sharma (anushar4)
    public Welcome() {
        options.add("1. Show Files");
        options.add("2. Show File Options Menu");
        options.add("3. Quit");
```

```java
        options.add("3. Quit");

    }
    1 usage  Anurag Sharma (anushar4) *
    public void introScreen() {
        System.out.println();
        System.out.println(welcomeText);
        System.out.println(developerText);
        System.out.println("============================");
        System.out.println();
        System.out.println("Directory : " + DirectoryService.getFileDirectory().name);
    }
    2 usages  Anurag Sharma (anushar4)
    @Override
    public void Show() {
        System.out.println();
        System.out.println("Main Menu");
        for (String s : options)  {
            System.out.println(s);
        }
    }
    2 usages  Anurag Sharma (anushar4)
    @Override
    public void NavigateOption(int option) {
        switch(option) {
```

```
            case 1:
                DirectoryService.PrintFiles();
                break;
            case 2:
                ScreenService.setCurrentScreen();
                break;
            case 3:
                System.out.println("\nThank You!!!");
            default:
                break;
        }
    }
    2 usages   Anurag Sharma (anushar4) *
    @Override
    public void GetUserInput() {
        int sch = 0;
        char ch;
        while(sch != 3) {
            this.Show();
            System.out.print("Enter the choice : ");
            ch = sc.next().charAt(0); //sch = sc.nextInt();
            sch = (int)ch - 48;
            this.NavigateOption(sch);
        }
    }
```

2: Directory Component

```
package Shared;

import java.io.File;
import java.nio.file.FileSystems;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.Collections;

4 usages   Anurag Sharma (anushar4)
public class Directory {

    2 usages
    public String name = "src/Storage/";
    4 usages
    private final ArrayList<File> files = new ArrayList<>();

    3 usages
    public Path path = FileSystems.getDefault().getPath(name).toAbsolutePath();
    1 usage
    File Dfiles = path.toFile();

    2 usages   Anurag Sharma (anushar4)
    public void fillFiles() {
        File[] directoryFiles = Dfiles.listFiles();
```

```
1 usage
File Dfiles = path.toFile();

2 usages  Anurag Sharma (anushar4)
public void fillFiles() {
    File[] directoryFiles = Dfiles.listFiles();

    files.clear();
    if (directoryFiles != null) {
        for (File directoryFile : directoryFiles)
            if (directoryFile.isFile())
                files.add(directoryFile);
    }

    Collections.sort(files);
}

1 usage  Anurag Sharma (anushar4)
public ArrayList<File> getFiles() {
    fillFiles();
    return files;
}
}
```

3: Screen Interface

```
package Shared;

import java.util.Scanner;

4 usages  2 implementations  Anurag Sharma (anushar4)
public interface Screen {

    2 usages
    Scanner sc = new Scanner(System.in);
    2 usages  2 implementations  Anurag Sharma (anushar4)
    void Show();
    2 usages  2 implementations  Anurag Sharma (anushar4)
    void NavigateOption(int option);
    2 usages  2 implementations  Anurag Sharma (anushar4)
    void GetUserInput();
}
```

4: File Component

A: File Options

```java
package FileMenu;

import Shared.Screen;

import java.util.ArrayList;

3 usages  Anurag Sharma (anushar4)
public class FileOptions implements Screen {

    5 usages
    private final ArrayList<String> options = new ArrayList<>();
    3 usages
    FileOperations file = new FileOperations();

    1 usage  Anurag Sharma (anushar4)
    public FileOptions() {
        options.add("1. Add A File");
        options.add("2. Delete A File");
        options.add("3. Search A File");
        options.add("4. Return to Menu");
    }
    2 usages  Anurag Sharma (anushar4)
    @Override
    public void Show() {
        System.out.println();
```

```java
        System.out.println();
        System.out.println("File Options Menu");
        for (String s : options) {
            System.out.println(s);
        }
    }
    2 usages  Anurag Sharma (anushar4)
    @Override
    public void NavigateOption(int option) {
        switch (option) {
            case 1:
                System.out.println("Adding File...");
                file.AddFile();
                break;
            case 2:
                System.out.println("Deleting File...");
                file.DeleteFile();
                break;
            case 3:
                System.out.println("Searching the file...");
                file.SearchFile();
                break;
        }
    }
    2 usages  Anurag Sharma (anushar4)
```

```
                file.DeleteFile();
                break;
            case 3:
                System.out.println("Searching the file...");
                file.SearchFile();
                break;
        }
    }
    2 usages  Anurag Sharma (anushar4)
    @Override
    public void GetUserInput() {
        char ch;
        int sch = 0;
        while (sch != 4) {
            this.Show();
            System.out.print("Enter the choice : ");
            ch = sc.next().charAt(0);
            //sch = sc.nextInt();
            sch = (int)ch - 48;
            this.NavigateOption(sch);
        }
    }
}
```

B: File Operations

```
package FileMenu;

import java.io.File;
import java.io.FileWriter;
import java.nio.file.Path;
import java.util.Scanner;

import Services.DirectoryService;

2 usages  Anurag Sharma (anushar4)
public class FileOperations {

    6 usages
    Scanner sc = new Scanner(System.in);
    5 usages
    String filename;
    3 usages
    char ch;

    1 usage  Anurag Sharma (anushar4)
    public void AddFile()   {
        System.out.print("Please Enter the file name with extension to Create : ");
        filename = sc.nextLine();

        File file = new File( pathname: DirectoryService.Path() + "/" + filename);
```

```
        File file = new File( pathname: DirectoryService.Path() + "/" + filename);
        try {
            boolean val = file.createNewFile();
            if(val) {
                System.out.println("File Created!!!");
                System.out.println("Do You Want to add Content");
                ch = sc.next().charAt(0);
                if(ch == 'Y' || ch == 'y') {
                    System.out.print("Enter the New Content : ");
                    sc.nextLine();
                    String fileData = sc.nextLine();
                    try {
                        FileWriter writeData = new FileWriter( fileName: DirectoryService.Path() + "/" + filename);
                        writeData.write(fileData);
                        System.out.println("Data is successfully added to the file.");
                        writeData.close();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
            else System.out.println("Unable to create");
        } catch(Exception e)    {
            e.printStackTrace();
        }
```

```
            else System.out.println("Unable to create");
        } catch(Exception e)    {
            e.printStackTrace();
        }
    }
    1 usage   Anurag Sharma (anushar4)
    public void DeleteFile()    {
        System.out.print("Please Enter the file name with extension to Delete : ");
        filename = sc.nextLine();

        File file = new File( pathname: DirectoryService.Path() + "/" + filename);
        try {
            boolean val = file.delete();
            if(val) System.out.println("File Deleted!!!");
            else System.out.println("Unable to Delete");

        } catch(Exception e)    {
            e.printStackTrace();
        }
    }
    1 usage   Anurag Sharma (anushar4)
    public void SearchFile() {
        boolean found = false;

        System.out.println("Please Enter the Filename:");
```

```
        boolean found = false;

        System.out.println("Please Enter the Filename:");
        String fileName = sc.nextLine();
        System.out.println("You are searching for a file named: " + fileName);

        Path path = DirectoryService.getFileDirectory().path;
        File Dfiles = path.toFile();

        File[] directoryFiles = Dfiles.listFiles();

        if (directoryFiles != null) {
            for (File directoryFile : directoryFiles)
                if (directoryFile.getName().equals(fileName)) {
                    System.out.println("Found " + fileName);
                    found = true;
                }
        }
        if (!found)
            System.out.println("File not found");
    }
}
```

5: Services Component

A: Directory Service

```
import Shared.Directory;

import java.io.File;

9 usages  Anurag Sharma (anushar4)
public class DirectoryService {
    3 usages
    private static final Directory fileDirectory = new Directory();

    1 usage  Anurag Sharma (anushar4)
    public static void PrintFiles() {
        fileDirectory.fillFiles();
        for (File file : DirectoryService.getFileDirectory().getFiles())
            System.out.println(file.getName());
    }
    3 usages  Anurag Sharma (anushar4)
    public static Directory getFileDirectory() { return fileDirectory; }
    3 usages  Anurag Sharma (anushar4)
    public static String Path() { return fileDirectory.path.toString();        // return fileDirectory.name; }
}
```

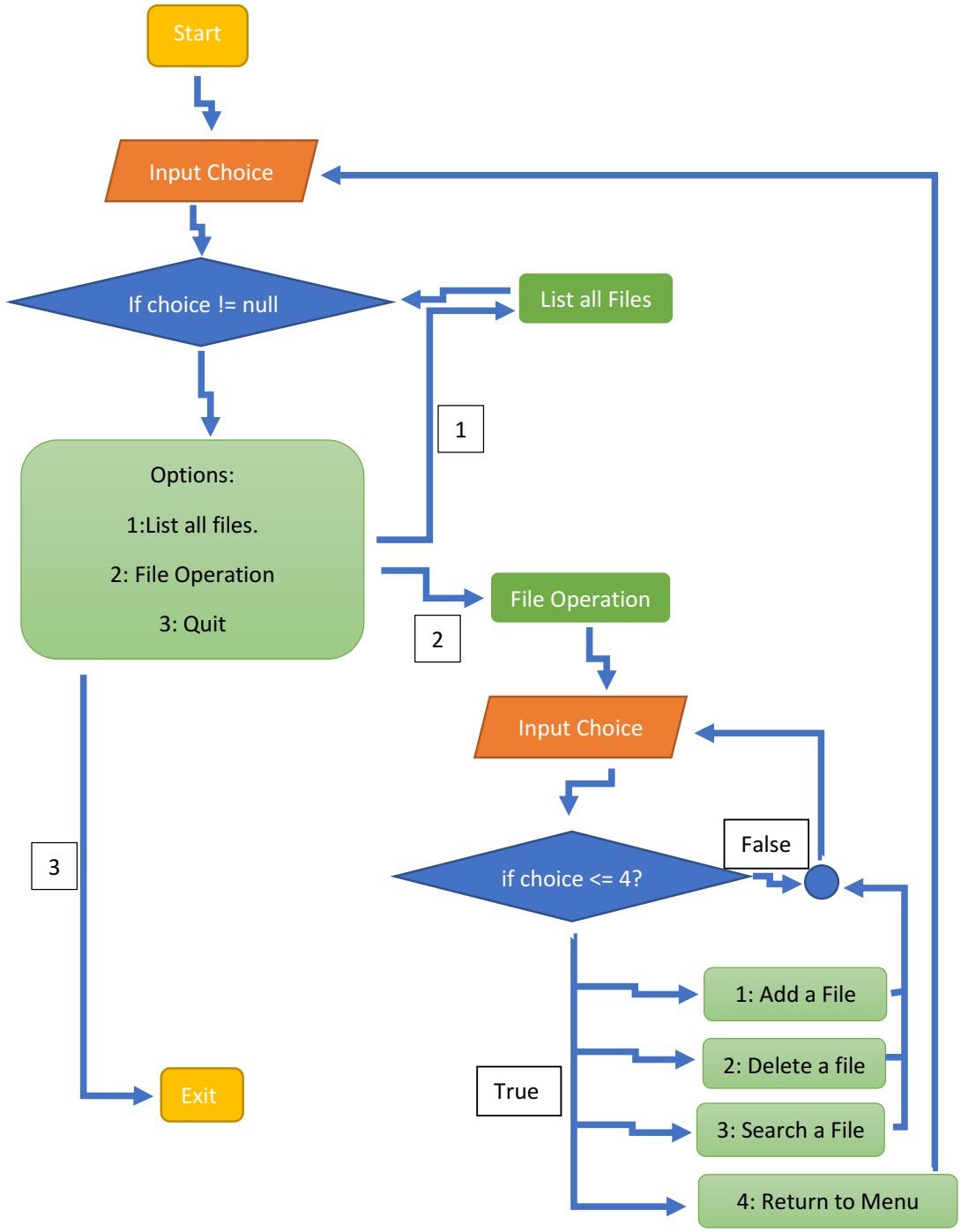
B: Screen Service

```
package Services;

import FileMenu.FileOptions;

2 usages  Anurag Sharma (anushar4)
public class ScreenService {
    1 usage  Anurag Sharma (anushar4)
    public static void setCurrentScreen()   {
        FileOptions file = new FileOptions();
        file.GetUserInput();
    }
}
```

6: Main/Core Component

```
import Welcome.Welcome;

2 usages  Anurag Sharma (anushar4)
public class Core {
    no usages  Anurag Sharma (anushar4)
    public static void main(String[] args)  {
        Welcome welcome = new Welcome();
        welcome.introScreen();
        welcome.GetUserInput();
    }
}
```

Flow Diagram



- Core Concepts used in this project are mostly basic Java libraries such as Class & Objects, Packages, Interfaces, Collections, ArrayList, Access specifier, Try-catch block, File Handling Concepts, Error Exception handling, Inheritance, abstract, final, static methods.

Algorithm

Step 1> Start

Step 2> input choice from the user.

Step 3> While choice != 3 then go to step 4.

Step 4> Switch(ch)

case 1: List all files in the specified directory and go back to step 2.

case 2: Go to step 5.

case 3: Go to Step 6.

default: Return back to step 2.

[End of switch case block]

[End of while loop]

Step 5> Input another choice sch from the user to perform file operations.

Step 5.1 > while loop sch != 4 then go to step 5.2.

Step 5.2> Switch(sch)

case 1: Add a file.

case 2: Delete a file.

case 3: Search a file.

case 4: Go to step 2.

default: Return back to step 5.

[End of switch case block]

[End of while loop]

Step 6> End the program.

Step 7> Stop

Conclusion

1: The prototype is robust and platform independent.

2: User can easily use the prototype and safely exit out of it.

3: The prototype has a good interface with CLI (Command Line Interface).

4: As a developer, we can enhance it by introducing several new features such as appending in a file or overwriting a file and the file details for which user selected.

5: This prototype though is robust but user can only interact it with terminal or CLI so we can develop a good GUI interface for more better user-friendly.

6: This prototype can also be implemented with multithreading to enable better performance.

7: And lastly, this prototype can be upgraded by implementing with authentication, validators, and securities patches to make it more versatile and secure in both local environment and global.