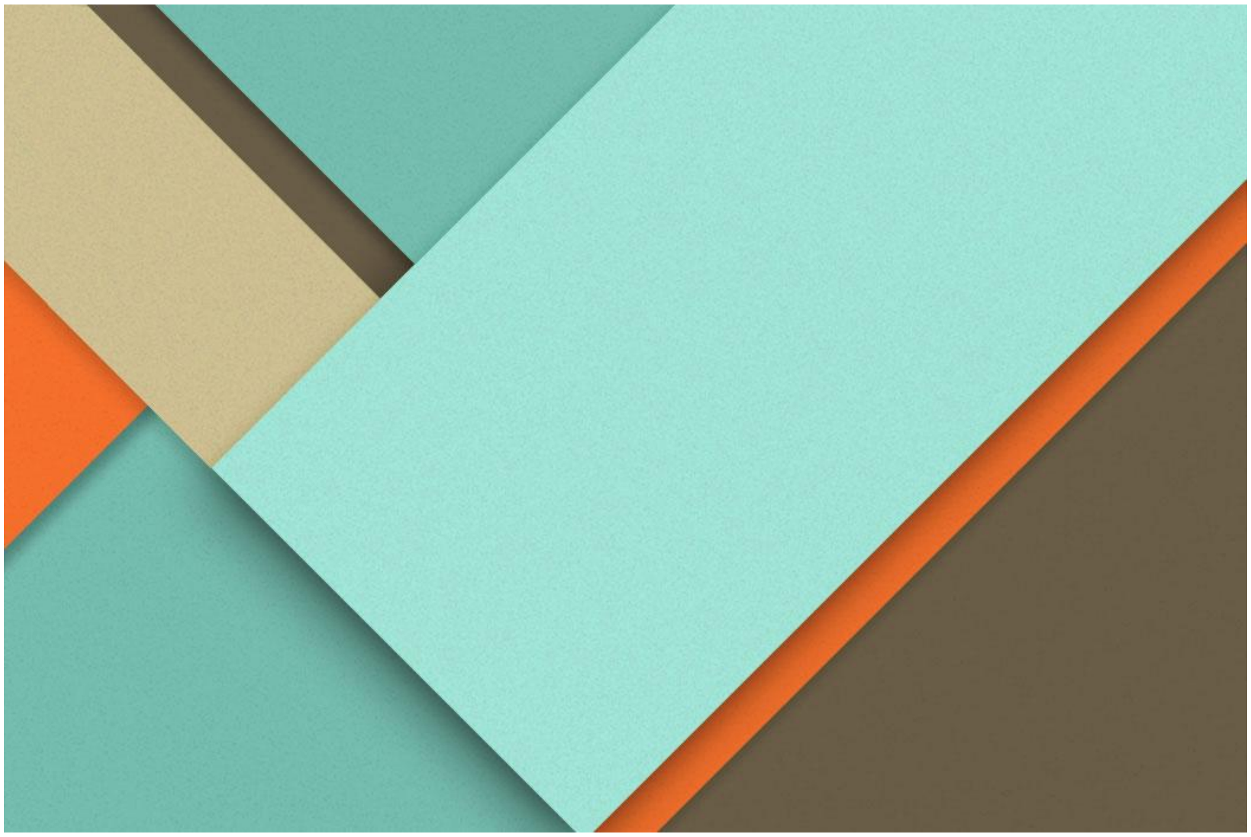


Project: LockedMe.com

Company Lockers Pvt. Ltd.

Vighnesh Gajula

Prototype of the application

Phase One Project

GitHub:-https://github.com/Vighnesh3232/PhaseOneProject

Overview

The application's prototype operates without a graphical user interface as a Command Line Interface (CLI). It is used to perform file operations like creating new files with content, deleting files, or searching for files in a particular directory before listing them in a sorting order.

The implementation is carried out with the assistance of IntelliJ's IDE and Java 19.0.1

Sprint Planning

The Execution is finished in two Sprint which are referenced beneath:

Sprint 1:

• Make the requirements and specifications clearer.

• Put the view content mechanism into use.

• Create a list of all files in alphabetical order.

• Implement features for a secure program termination.

Sprint 2:

• Add the capability to create files alongside the content.

• Include a way to delete a file if it's in a directory specified by the user.

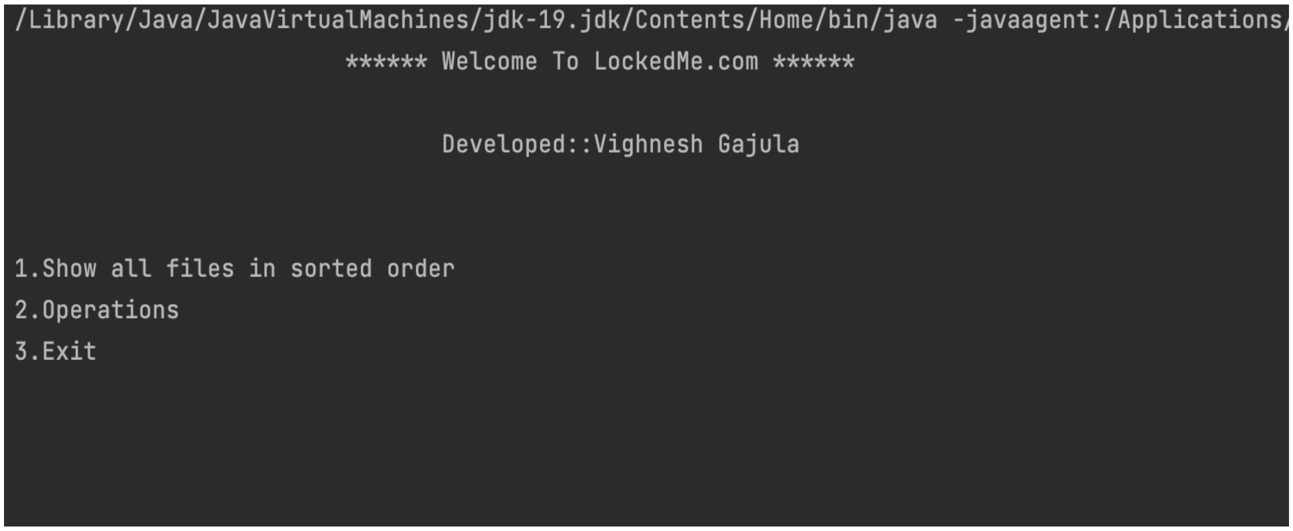
• Carry out usefulness to look through a document in a similar catalog.

• Documentation

DOCUMENTATION OF THE FUNCTIONALITY

The various options from which the user can select specific file operations are listed here.

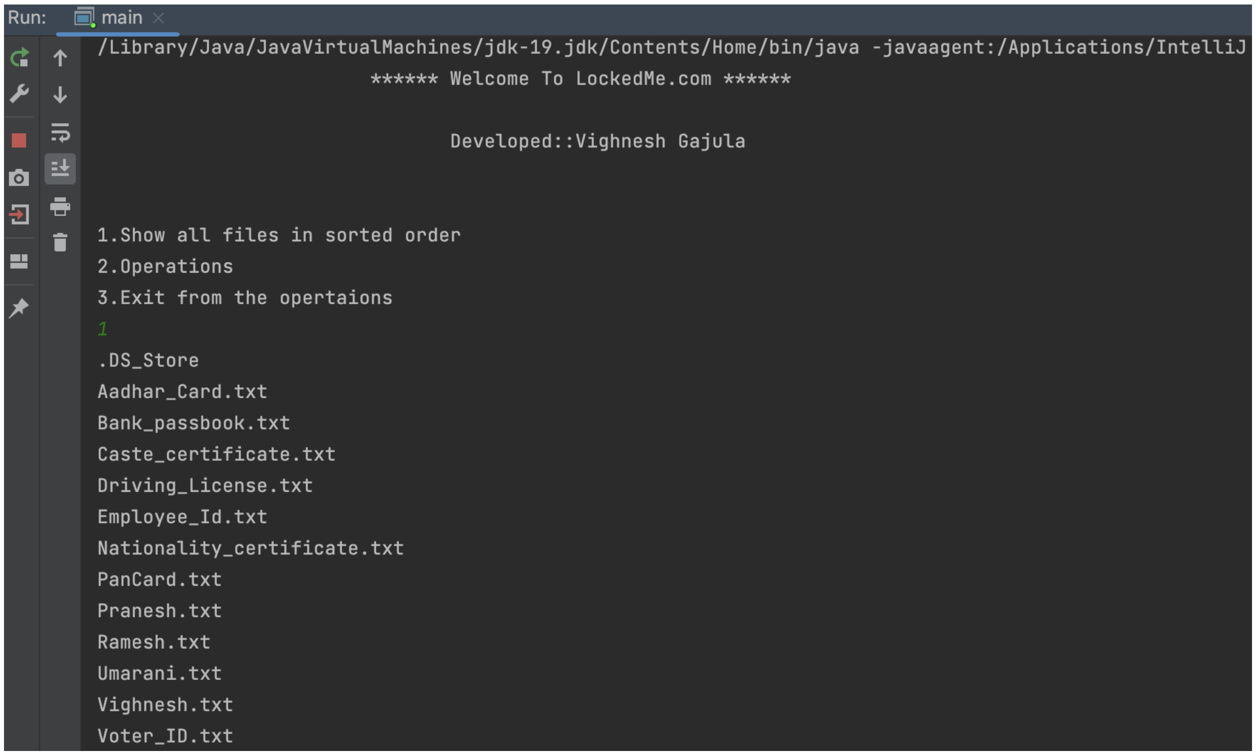
This the FrontScreen which user interacts



```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/
                        ****** Welcome To LockedMe.com ******

                               Developed::Vighnesh Gajula


1.Show all files in sorted order
2.Operations
3.Exit
```

Option 1) Show all files in sorted order.

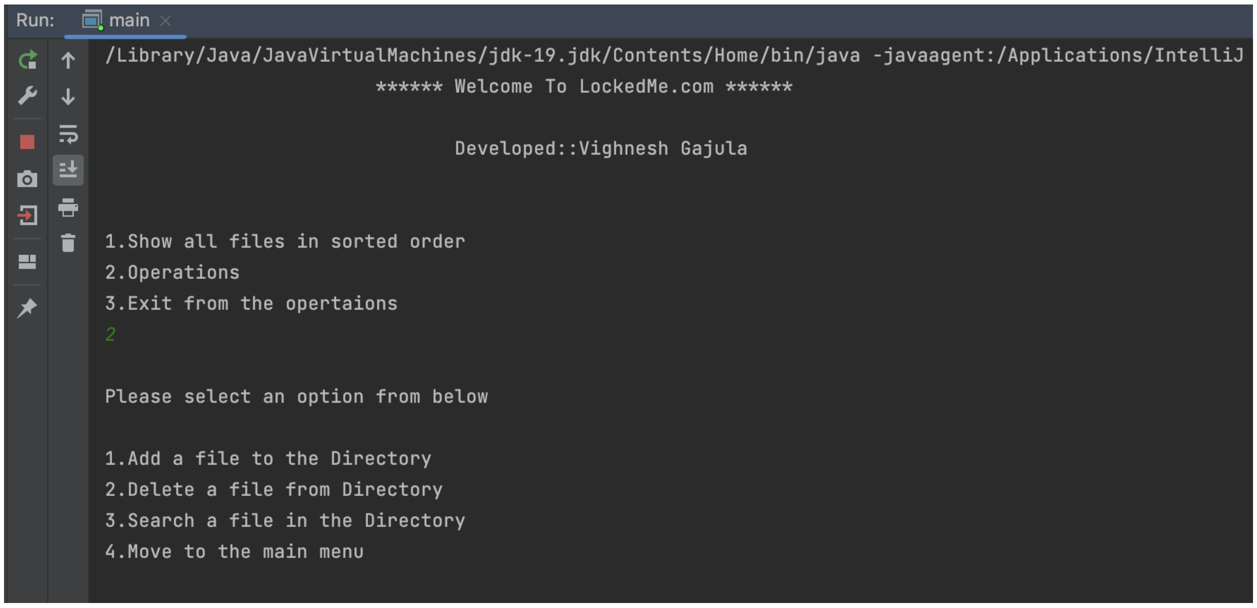
This option will let the user see a list of files in the specified directory in sorted order.



Run: main

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
                        ****** Welcome To LockedMe.com ******

                               Developed::Vighnesh Gajula



1.Show all files in sorted order
2.Operations
3.Exit from the opertaions
1
.DS_Store
Aadhar_Card.txt
Bank_passbook.txt
Caste_certificate.txt
Driving_License.txt
Employee_Id.txt
Nationality_certificate.txt
PanCard.txt
Pranesh.txt
Ramesh.txt
Umarani.txt
Vighnesh.txt
Voter_ID.txt
```

Option 2) Operations:



Run: main

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
                        ****** Welcome To LockedMe.com ******

                               Developed::Vighnesh Gajula



1.Show all files in sorted order
2.Operations
3.Exit from the opertaions
2

Please select an option from below

1.Add a file to the Directory
2.Delete a file from Directory
3.Search a file in the Directory
4.Move to the main menu
```

This option will let users provide several file operations with.

The Operations command gives us a few operations to perform such as:

1.Add a file to the Directory

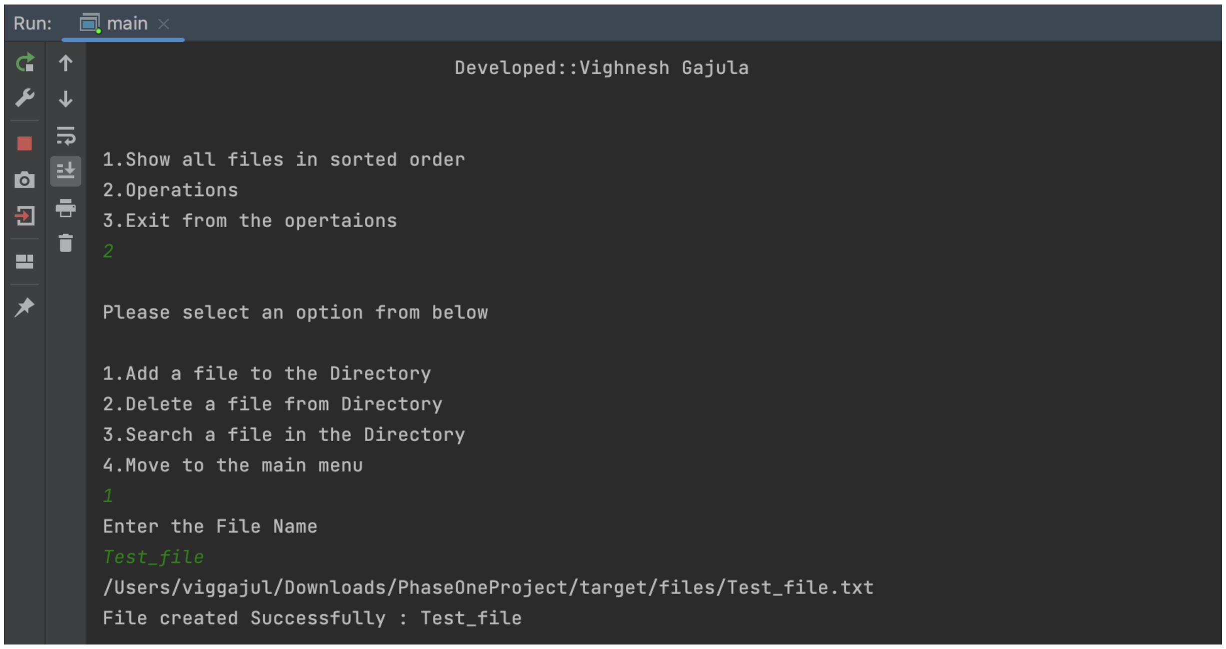
2.Delete a file from Directory

3.Search a file in the Directory

4.Move to the main menu

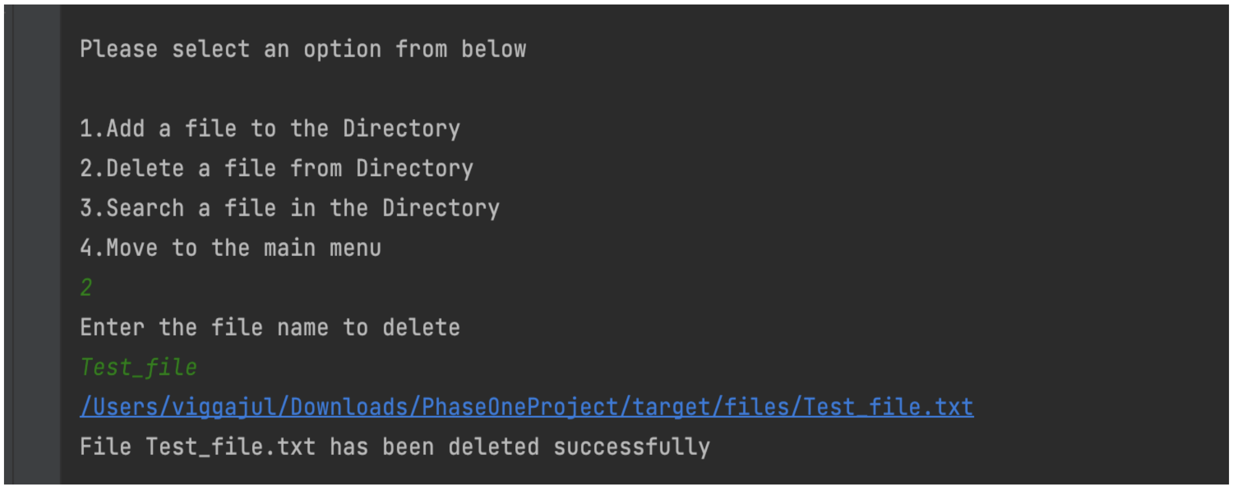
For the operations in Operations command.

1.Add a file to the Directory:- Creates File



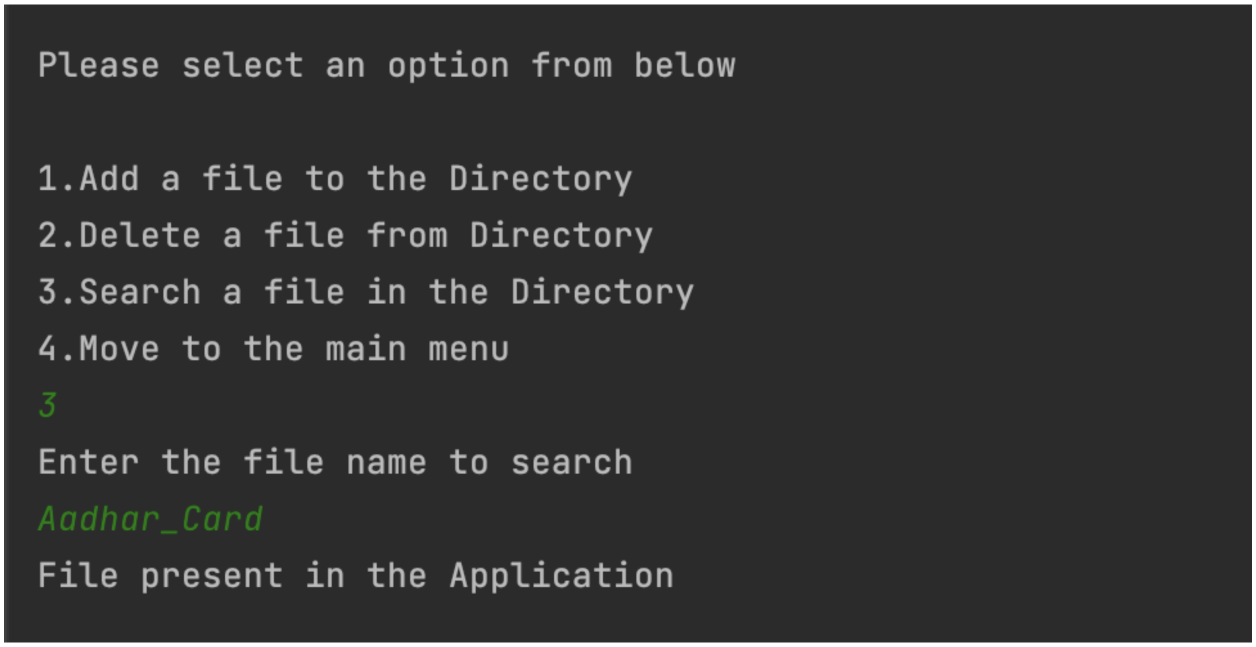
```
                    Developed::Vighnesh Gajula


1.Show all files in sorted order
2.Operations
3.Exit from the opertaions
2

Please select an option from below

1.Add a file to the Directory
2.Delete a file from Directory
3.Search a file in the Directory
4.Move to the main menu
1
Enter the File Name
Test_file
/Users/viggajul/Downloads/PhaseOneProject/target/files/Test_file.txt
File created Successfully : Test_file
```

2.Delete a file from Directory: This command will delete the created file.



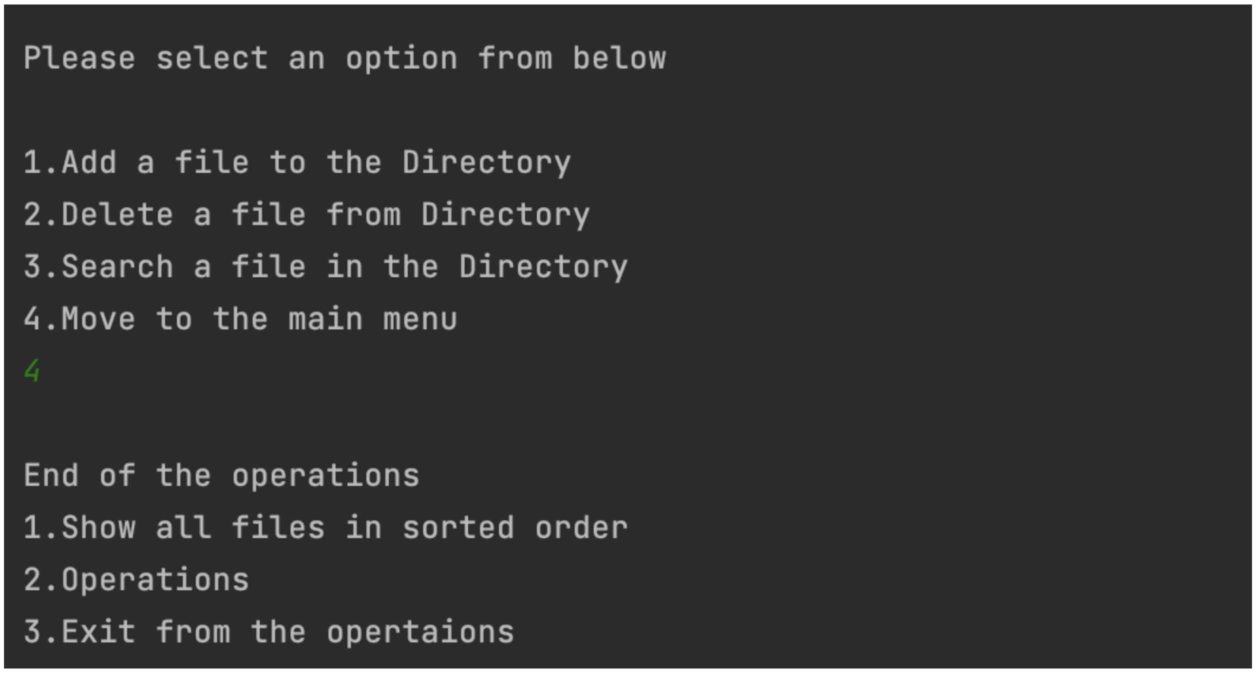
```
Please select an option from below

1.Add a file to the Directory
2.Delete a file from Directory
3.Search a file in the Directory
4.Move to the main menu
2
Enter the file name to delete
Test_file
/Users/viggajul/Downloads/PhaseOneProject/target/files/Test_file.txt
File Test_file.txt has been deleted successfully
```

3.Search a file in the Directory: This command will search the created file.



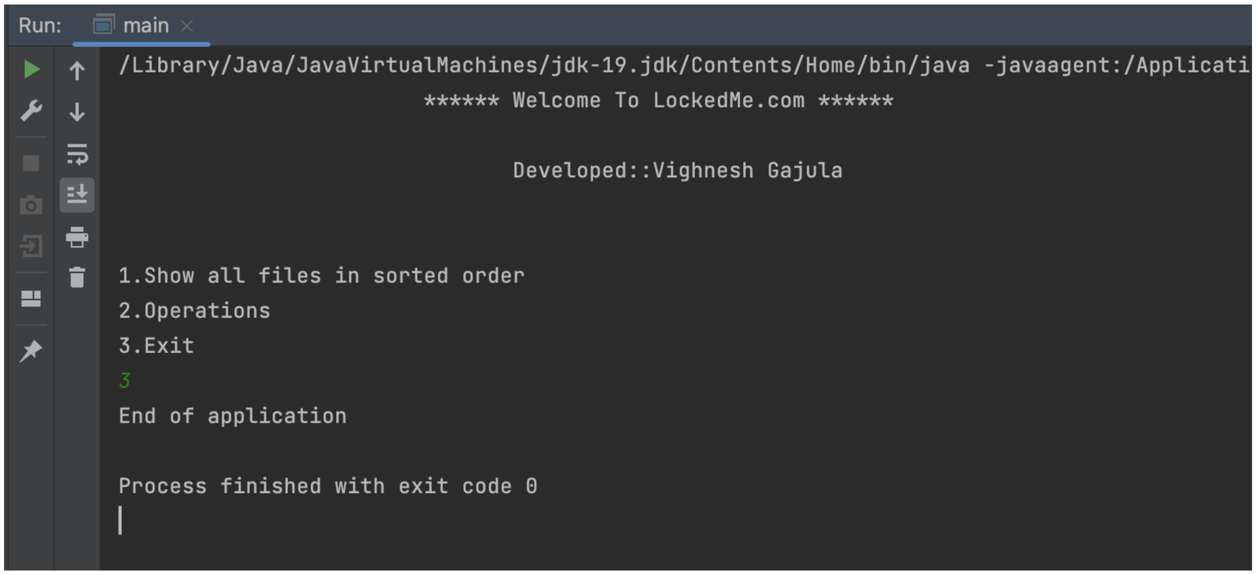
```
Please select an option from below

1.Add a file to the Directory
2.Delete a file from Directory
3.Search a file in the Directory
4.Move to the main menu
3
Enter the file name to search
Aadhar_Card
File present in the Application
```

4.Move to the main menu: This command will exit back from the Operations.



```
Please select an option from below

1.Add a file to the Directory
2.Delete a file from Directory
3.Search a file in the Directory
4.Move to the main menu
4


End of the operations
1.Show all files in sorted order
2.Operations
3.Exit from the opertaions
```

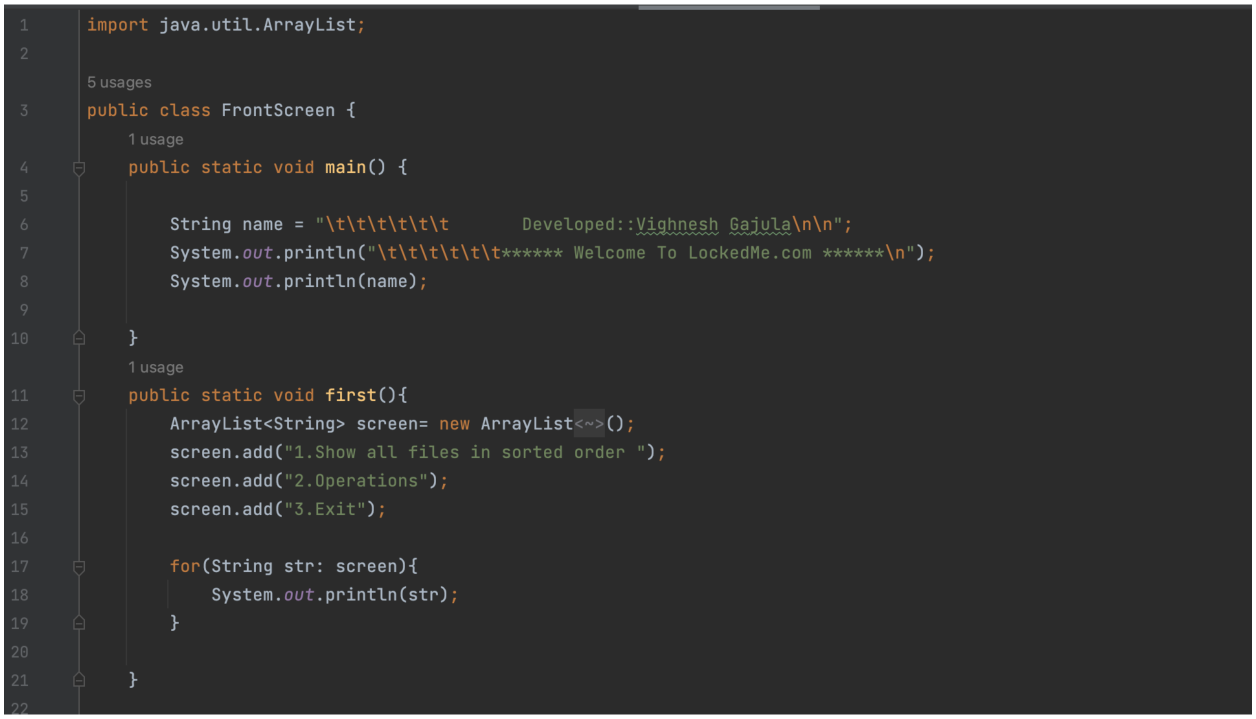
Option 3) Exit:- This will permit clients to exit from the program securely.



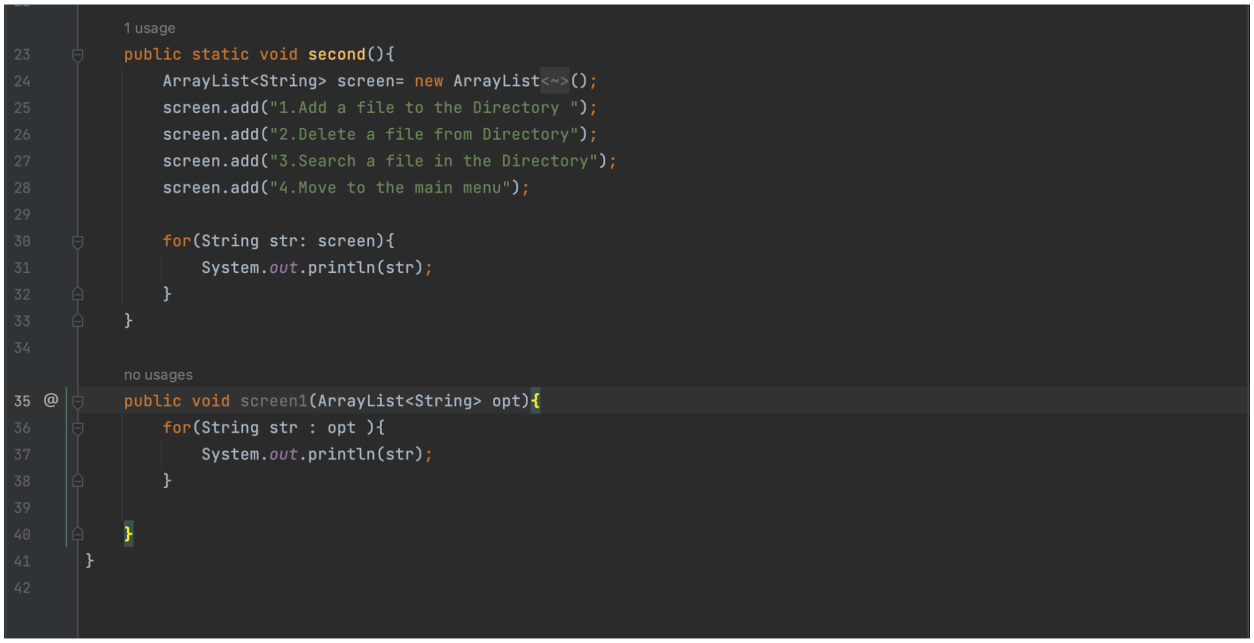
```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applicati
                         ****** Welcome To LockedMe.com ******

                               Developed::Vighnesh Gajula



1.Show all files in sorted order
2.Operations
3.Exit
3
End of application

Process finished with exit code 0
```

SOURCE CODE

1) Front Screen / welcome screen

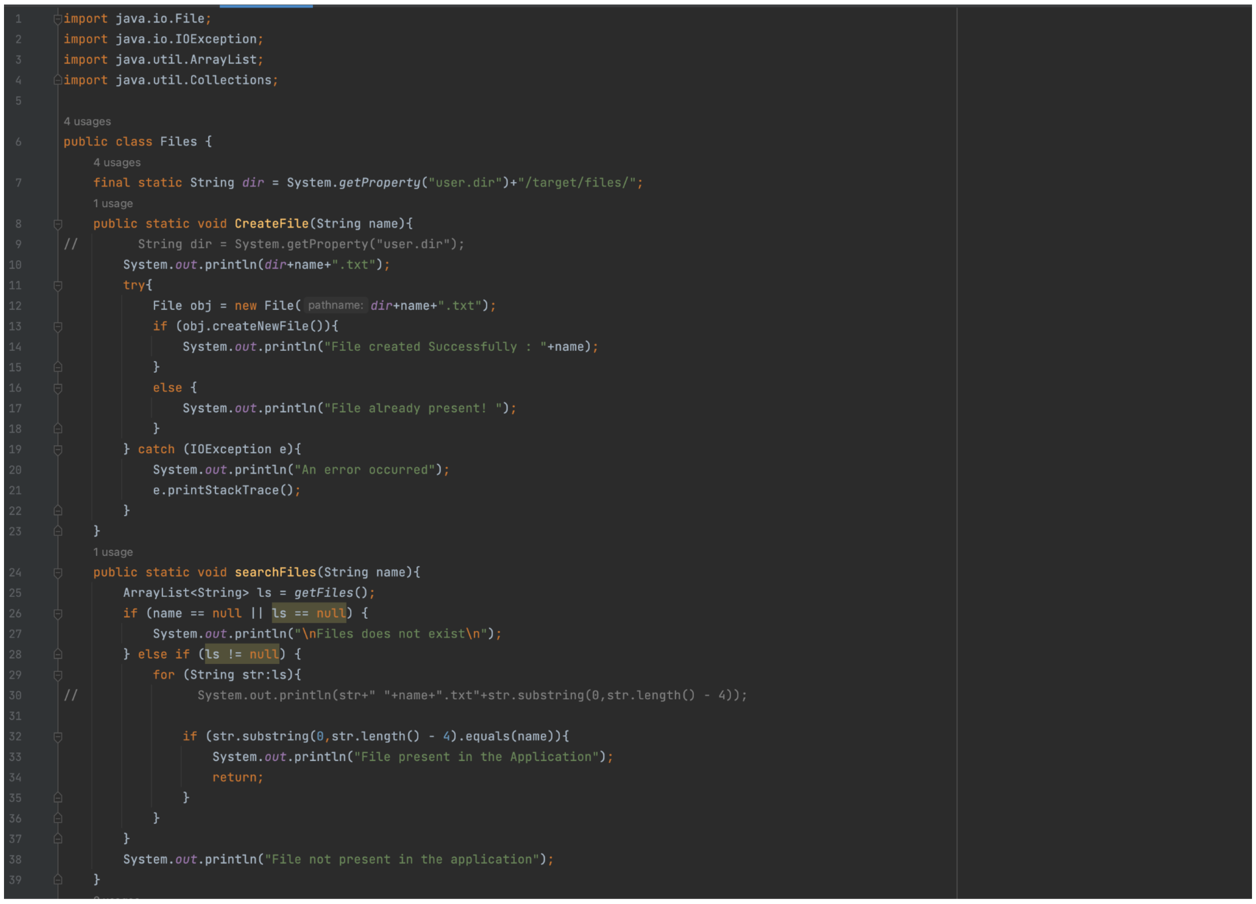


```java
import java.util.ArrayList;

5 usages
public class FrontScreen {
    1 usage
    public static void main() {

        String name = "\t\t\t\t\t\t       Developed::Vighnesh Gajula\n\n";
        System.out.println("\t\t\t\t\t\t****** Welcome To LockedMe.com ******\n");
        System.out.println(name);

    }
    1 usage
    public static void first(){
        ArrayList<String> screen= new ArrayList<~>();
        screen.add("1.Show all files in sorted order ");
        screen.add("2.Operations");
        screen.add("3.Exit");

        for(String str: screen){
            System.out.println(str);
        }

    }
```

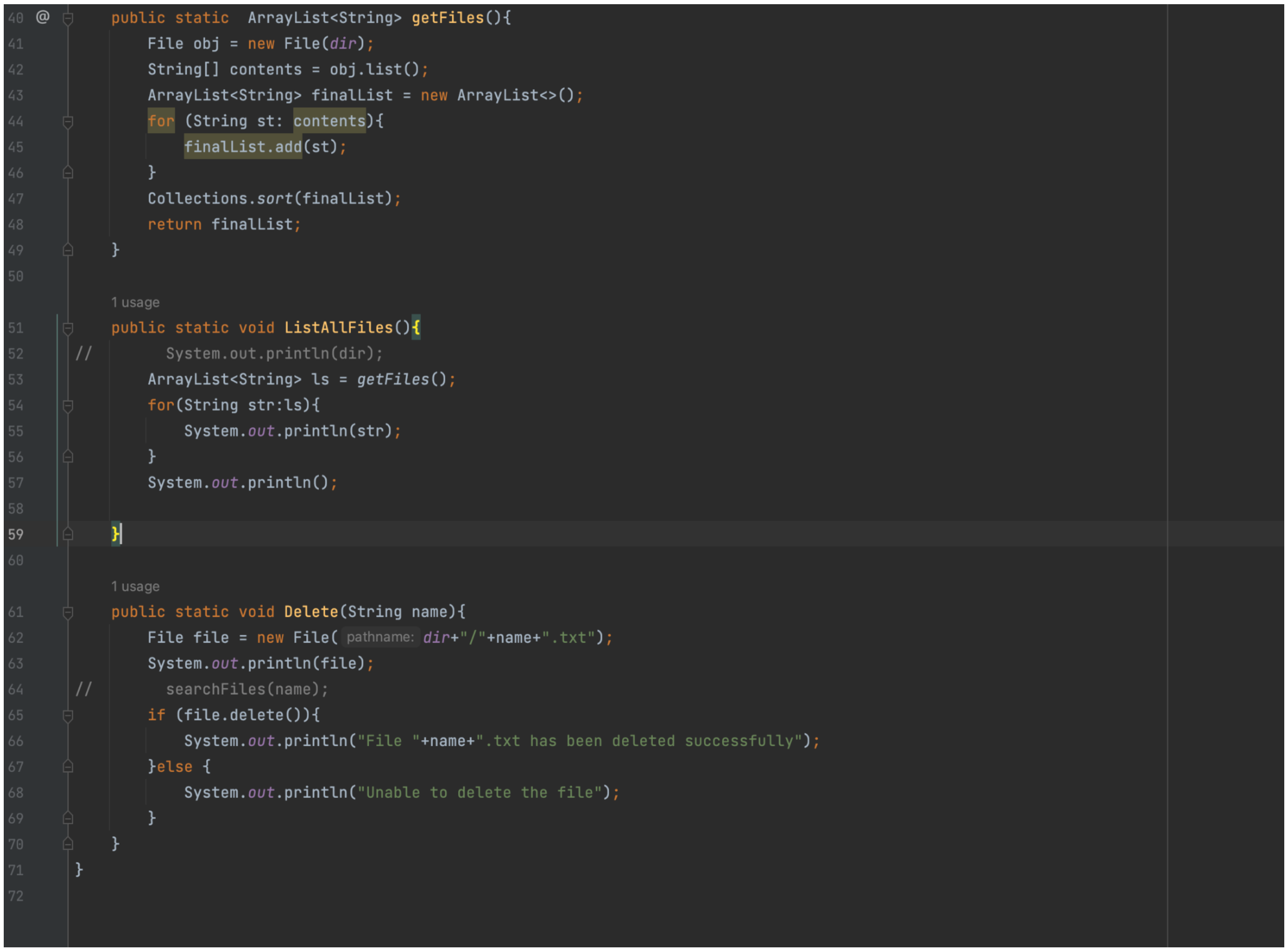


```java
    1 usage
    public static void second(){
        ArrayList<String> screen= new ArrayList<~>();
        screen.add("1.Add a file to the Directory ");
        screen.add("2.Delete a file from Directory");
        screen.add("3.Search a file in the Directory");
        screen.add("4.Move to the main menu");

        for(String str: screen){
            System.out.println(str);
        }
    }

    no usages
    public void screen1(ArrayList<String> opt){
        for(String str : opt ){
            System.out.println(str);
        }

    }
}
```

2) Files component

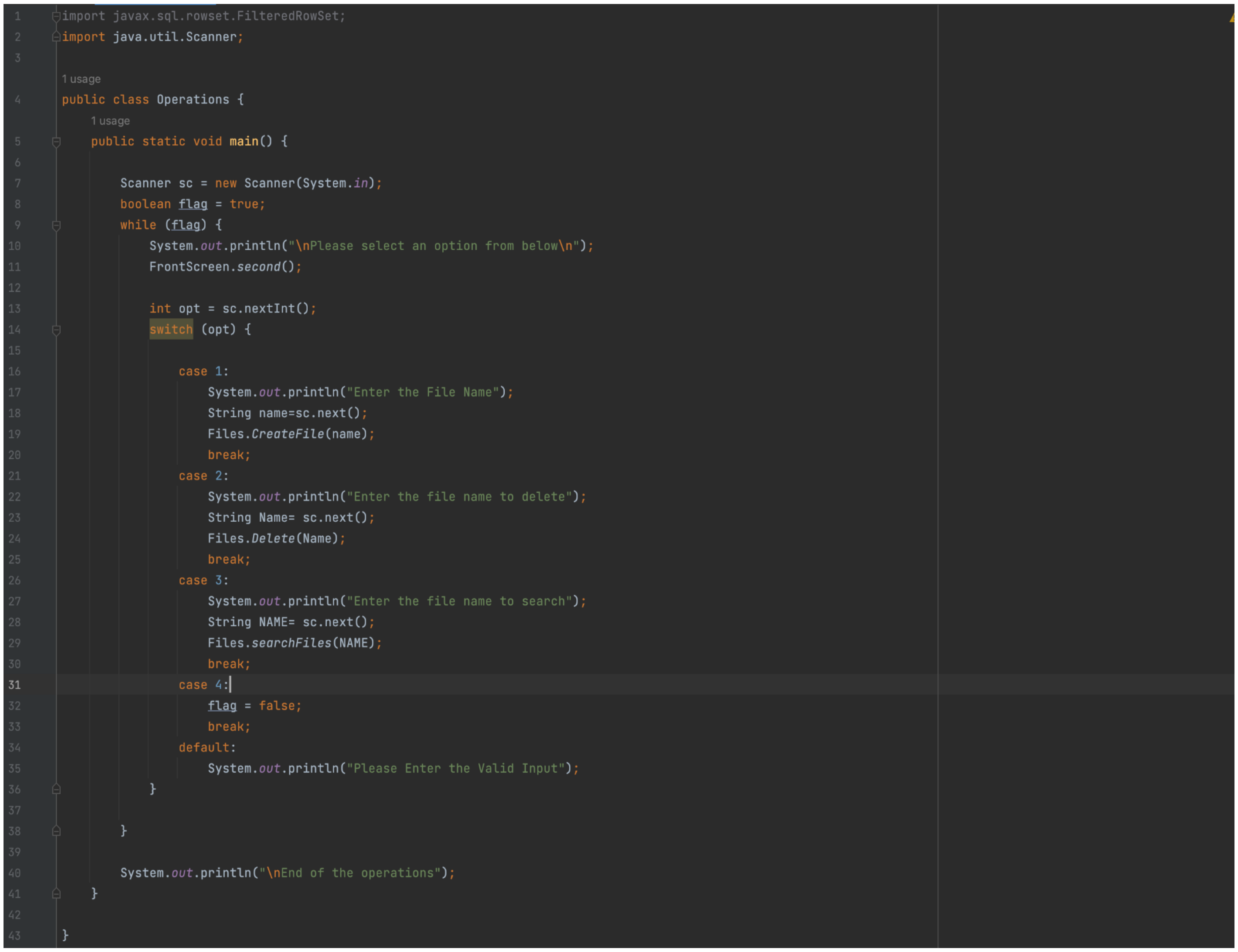


```java
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

4 usages
public class Files {
    4 usages
    final static String dir = System.getProperty("user.dir")+"/target/files/";
    1 usage
    public static void CreateFile(String name){
//        String dir = System.getProperty("user.dir");
        System.out.println(dir+name+".txt");
        try{
            File obj = new File( pathname: dir+name+".txt");
            if (obj.createNewFile()){
                System.out.println("File created Successfully : "+name);
            }
            else {
                System.out.println("File already present! ");
            }
        } catch (IOException e){
            System.out.println("An error occurred");
            e.printStackTrace();
        }
    }
    1 usage
    public static void searchFiles(String name){
        ArrayList<String> ls = getFiles();
        if (name == null || ls == null) {
            System.out.println("\nFiles does not exist\n");
        } else if (ls != null) {
            for (String str:ls){
//                System.out.println(str+" "+name+".txt"+str.substring(0,str.length() - 4));

                if (str.substring(0,str.length() - 4).equals(name)){
                    System.out.println("File present in the Application");
                    return;
                }
            }
        }
        System.out.println("File not present in the application");
    }
```



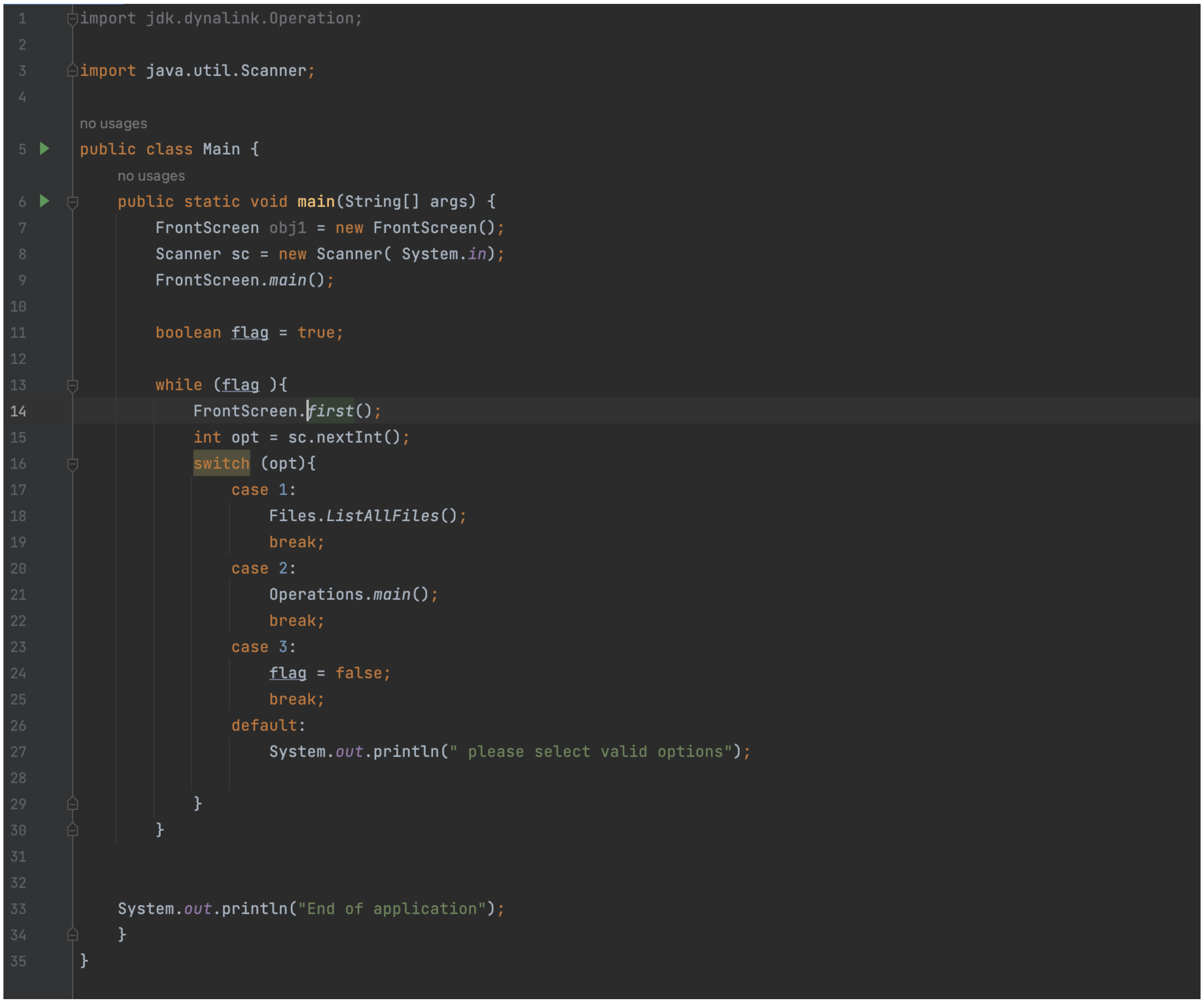
```
    public static  ArrayList<String> getFiles(){
        File obj = new File(dir);
        String[] contents = obj.list();
        ArrayList<String> finalList = new ArrayList<>();
        for (String st: contents){
            finalList.add(st);
        }
        Collections.sort(finalList);
        return finalList;
    }

    1 usage
    public static void ListAllFiles(){
//        System.out.println(dir);
        ArrayList<String> ls = getFiles();
        for(String str:ls){
            System.out.println(str);
        }
        System.out.println();

    }

    1 usage
    public static void Delete(String name){
        File file = new File( pathname: dir+"/"+name+".txt");
        System.out.println(file);
//        searchFiles(name);
        if (file.delete()){
            System.out.println("File "+name+".txt has been deleted successfully");
        }else {
            System.out.println("Unable to delete the file");
        }
    }
}
```

3) Operations component



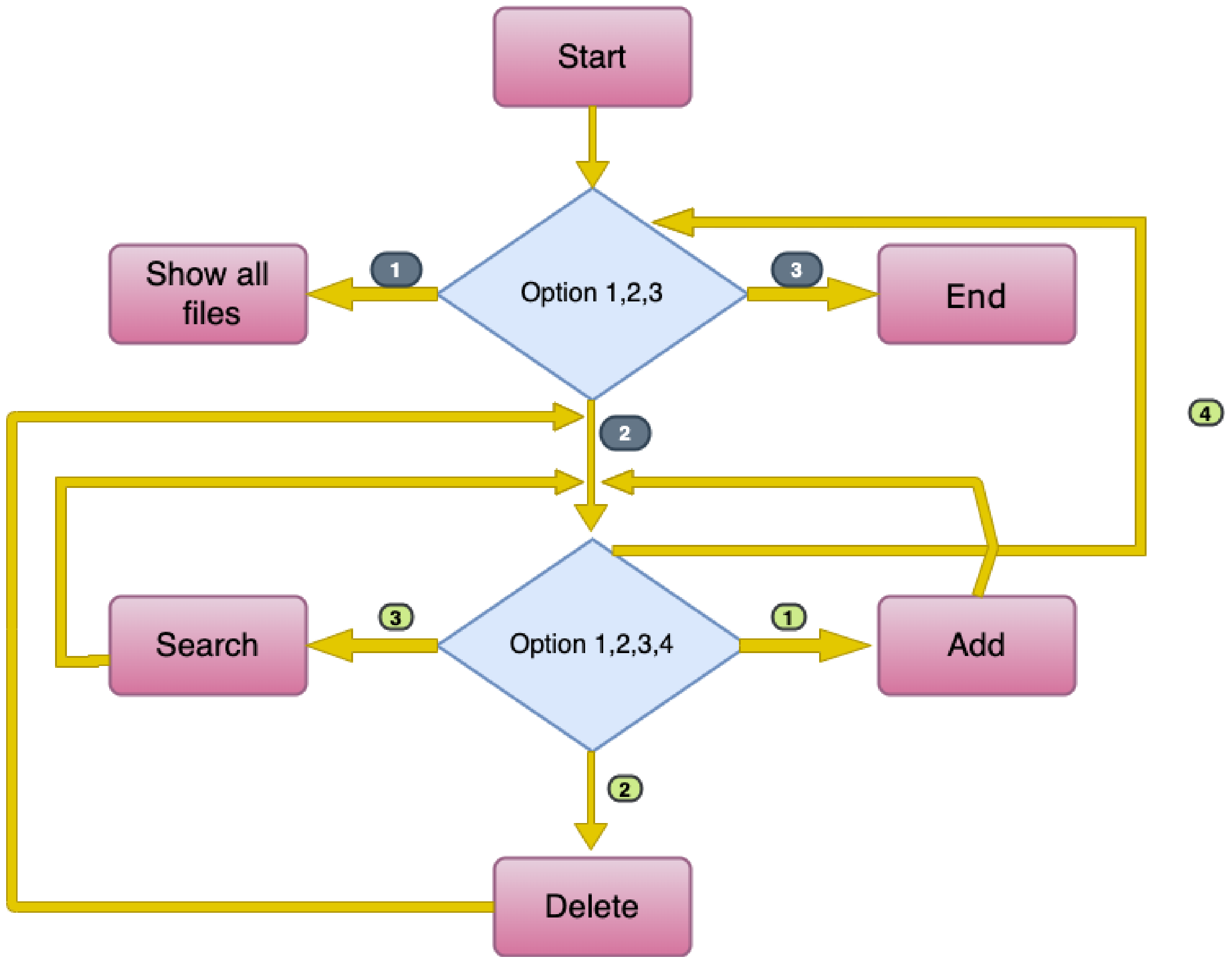
```java
import javax.sql.rowset.FilteredRowSet;
import java.util.Scanner;

1 usage
public class Operations {
    1 usage
    public static void main() {

        Scanner sc = new Scanner(System.in);
        boolean flag = true;
        while (flag) {
            System.out.println("\nPlease select an option from below\n");
            FrontScreen.second();

            int opt = sc.nextInt();
            switch (opt) {

                case 1:
                    System.out.println("Enter the File Name");
                    String name=sc.next();
                    Files.CreateFile(name);
                    break;
                case 2:
                    System.out.println("Enter the file name to delete");
                    String Name= sc.next();
                    Files.Delete(Name);
                    break;
                case 3:
                    System.out.println("Enter the file name to search");
                    String NAME= sc.next();
                    Files.searchFiles(NAME);
                    break;
                case 4:
                    flag = false;
                    break;
                default:
                    System.out.println("Please Enter the Valid Input");
            }

        }

        System.out.println("\nEnd of the operations");
    }

}
```

4) Main Component



```java
import jdk.dynalink.Operation;

import java.util.Scanner;

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        FrontScreen obj1 = new FrontScreen();
        Scanner sc = new Scanner( System.in);
        FrontScreen.main();

        boolean flag = true;

        while (flag ){
            FrontScreen.first();
            int opt = sc.nextInt();
            switch (opt){
                case 1:
                    Files.ListAllFiles();
                    break;
                case 2:
                    Operations.main();
                    break;
                case 3:
                    flag = false;
                    break;
                default:
                    System.out.println(" please select valid options");

            }
        }


    System.out.println("End of application");
    }
}
```

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 Client.
Step 3- While Flag != False at that point go to step 4.
Step 4- Switch(opt)
            case 1:List all files within the indicated registry and then go back to step 2.
            case 2: Go to step 5.
            default: Return back to step 2.
            [End of switch case block]
 [End of while loop]
 Step 5- Input another choice opt from the user to perform file operations.
            Step 5.1 - Flag != False then go to step 5.2.
            Step 5.2 - Switch(opt)
                        case 1:Add a file to the Directory.
                        case 2: Delete a file from Directory
                        case 3:Search a file in the Directory.
                        case 4: Move to the main menu which sends 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 model is strong and stage autonomous.

2: Clients can effectively utilize the model and securely exit out of it.

3: The model includes a great interface with CLI (Command Line Interface).

4: As an engineer, we will upgrade it by presenting a few modern highlights such as adding in a record or overwriting a record and the record points of interest for which the client selected.

5: This model in spite of the fact that it is strong but client can as it were associated with terminal or CLI so we can create a great GUI interface for more superior user-friendly.

6: This model can moreover be executed with multithreading to empower way 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.