JAVA FSD PHASE 1 ASSIGNMENT

PROJECT DETAILS:

EMPLOYEE ID : 2527463

EMPLOYEE NAME : Vishnu R

PROJECT DETAILS: JAVA FSD Phase 1assessment module.

To develop a prototype application module for

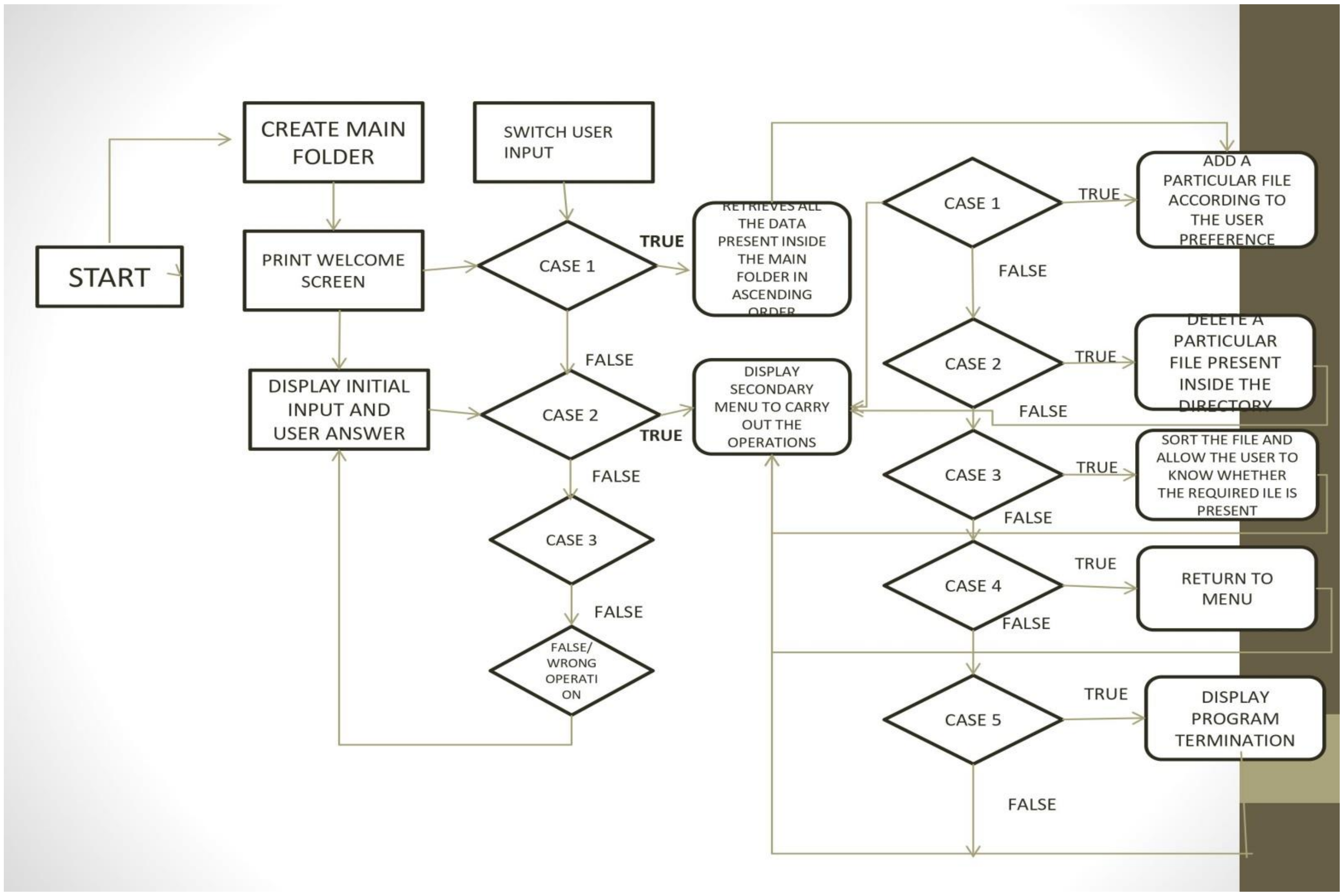
LOCKEDME.com Company.

SPRINTS AND TASKS:

The task is planned to be completed in a single sprint. The tasks to be completed are

- Creating flow of application.
- Initializing git repository to track changes as the application progress.
- Writing and compiling java program to fulfill the required condition.
- Testing and Dry run the application developed to meet the user requirement.
- Push the Code into github.
- Specifying application details.

ALGORITHM AND FLOWCHART:


START
CREATE MAIN FOLDER
PRINT WELCOME SCREEN
DISPLAY INITIAL INPUT AND USER ANSWER
SWITCH USER INPUT
CASE 1
TRUE
FALSE
CASE 2
TRUE
FALSE
CASE 3
FALSE
FALSE/ WRONG OPERATI ON
RETRIEVES ALL THE DATA PRESENT INSIDE THE MAIN FOLDER IN ASCENDING ORDER
DISPLAY SECONDARY MENU TO CARRY OUT THE OPERATIONS
CASE 1
TRUE
ADD A PARTICULAR FILE ACCORDING TO THE USER PREFERENCE
FALSE
CASE 2
TRUE
DELETE A PARTICULAR FILE PRESENT INSIDE THE DIRECTORY
FALSE
CASE 3
TRUE
SORT THE FILE AND ALLOW THE USER TO KNOW WHETHER THE REQUIRED ILE IS PRESENT
FALSE
CASE 4
TRUE
RETURN TO MENU
FALSE
CASE 5
TRUE
DISPLAY PROGRAM TERMINATION
FALSE

THE ABOVE FLOWCHART DIAGRAM EXPLAINS THE WORKING FLOW OF THE PROGRAM.

ALGORITHM/ PROGRAM BUILDING:

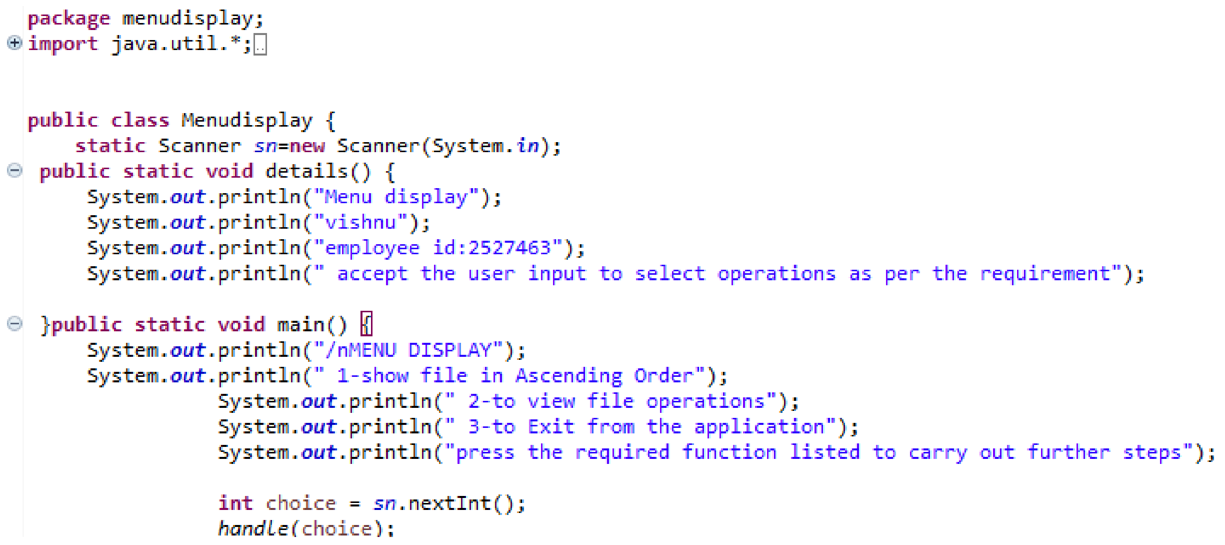
TO DEMONSTRATE THE CONCEPT USED WE ARE STATING THE ALGORITHMIC CONCEPT TO DESIGN THE PROGRAM.

- ❖ CREATE THE PROJECT IN ECLIPSE.
- ❖ WRITE A PROGRAM TO ALLOW THE ENTRY POINT OR WELCOME PAGE.
- ❖ WRITE PROGRAM TO CARRY OUT THE ENTIRE USER DEFINED OPERATIONS.
- ❖ RUN THE TEST CASE SCENARIOS.
- ❖ UPLOAD THE CONTENT INTO GITHUB LINK.

Step 1: Creating a new project in Eclipse

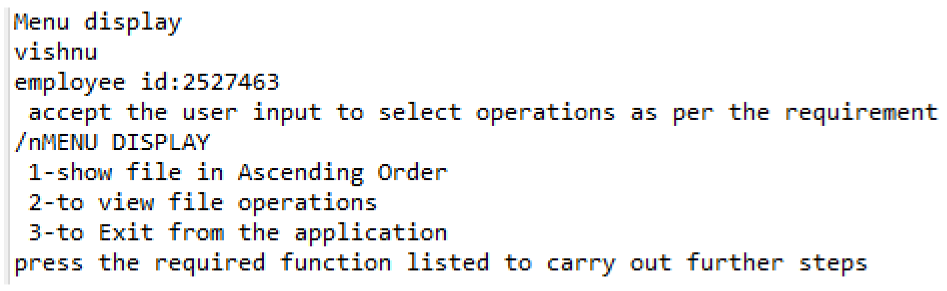
- Open Eclipse
- Go to File -> New -> Project -> Java Project -> Next.
- Type in any project name and click on “Finish.”
- Select your project and go to File -> New -> Class.
- Enter **Menudisplay** in any class name, check the checkbox “public static void main (String [] args)”, and click on “Finish.”

Step 2: WRITE A PROGRAM TO ALLOW THE ENTRY POINT OR WELCOME PAGE



```
package menudisplay;
import java.util.*;


public class Menudisplay {
    static Scanner sn=new Scanner(System.in);
 public static void details() {
     System.out.println("Menu display");
     System.out.println("vishnu");
     System.out.println("employee id:2527463");
     System.out.println(" accept the user input to select operations as per the requirement");

 }public static void main() {
     System.out.println("/nMENU DISPLAY");
     System.out.println(" 1-show file in Ascending Order");
                System.out.println(" 2-to view file operations");
                System.out.println(" 3-to Exit from the application");
                System.out.println("press the required function listed to carry out further steps");

                int choice = sn.nextInt();
                handle(choice);
```

This Particular Program Displays The Employee Details And Menu Options That We Have Provided.



```
Menu display
vishnu
employee id:2527463
 accept the user input to select operations as per the requirement
/nMENU DISPLAY
 1-show file in Ascending Order
 2-to view file operations
 3-to Exit from the application
press the required function listed to carry out further steps
```

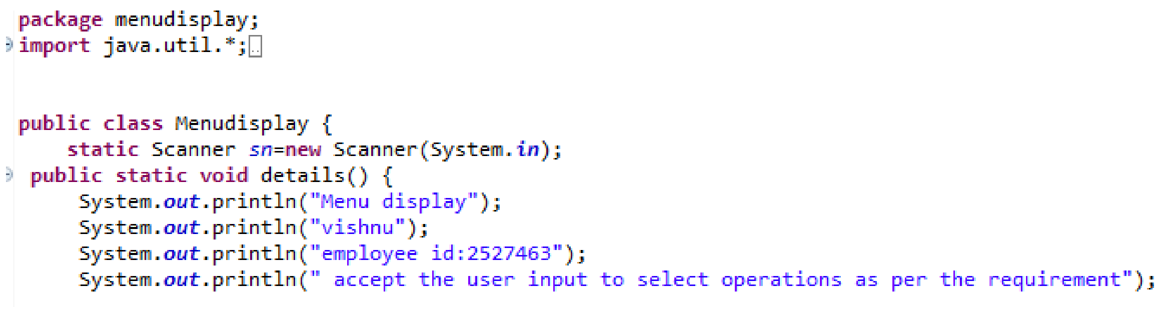
We have to enable the packages to be imported in the Menudisplay class so that the class could be extended and the required operations could be carried out.

Step 3: WRITE PROGRAM TO CARRY OUT THE ENTIRE USER DEFINED OPERATIONS.

The following data's should be taken into consideration while designing the program.

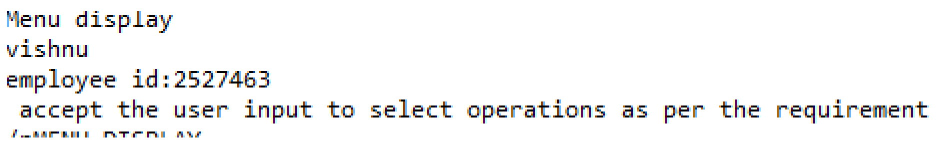
1. Display welcome page with details.
2. Enable Initial menu options.
3. Provide the secondary operations and menu for user defined actions.

Display welcome page with details.



```
package menudisplay;
import java.util.*;


public class Menudisplay {
    static Scanner sn=new Scanner(System.in);
 public static void details() {
     System.out.println("Menu display");
     System.out.println("vishnu");
     System.out.println("employee id:2527463");
     System.out.println(" accept the user input to select operations as per the requirement");
```

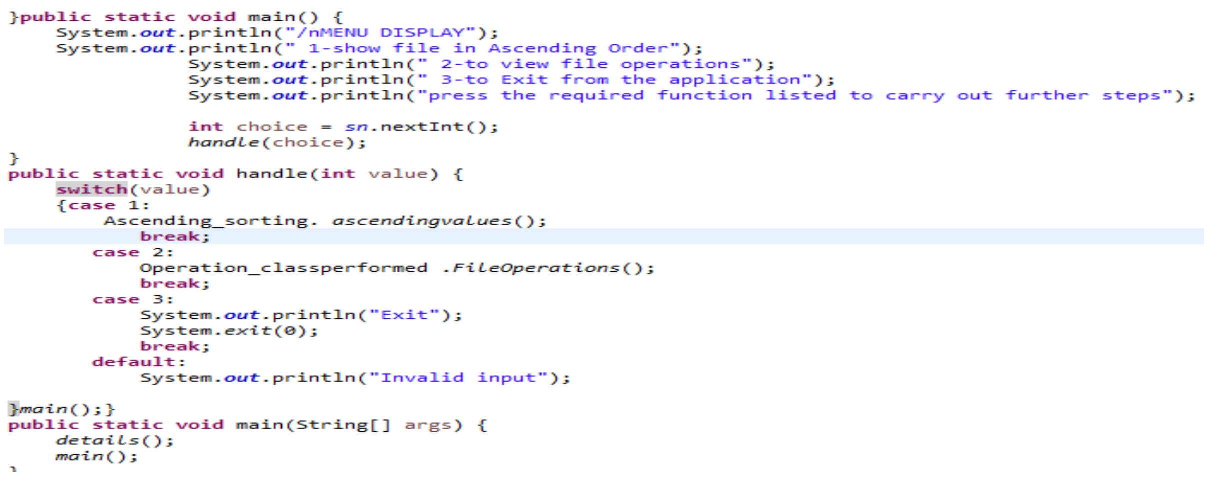
Output:



```
Menu display
vishnu
employee id:2527463
 accept the user input to select operations as per the requirement
```

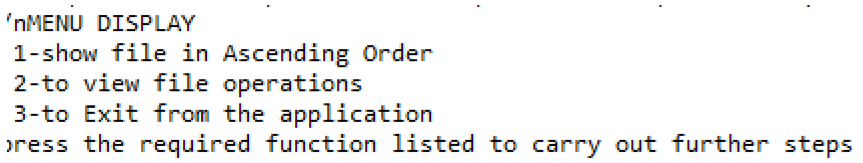
Particular data to be shown before the initial menu processing to take place.

Enable Initial menu options.



```
}public static void main() {
    System.out.println("/nMENU DISPLAY");
    System.out.println(" 1-show file in Ascending Order");
                System.out.println(" 2-to view file operations");
                System.out.println(" 3-to Exit from the application");
                System.out.println("press the required function listed to carry out further steps");

                int choice = sn.nextInt();
                handle(choice);
}
public static void handle(int value) {
    switch(value)
    {case 1:
        Ascending_sorting. ascendingvalues();
            break;
        case 2:
            Operation_classperformed .FileOperations();
            break;
        case 3:
            System.out.println("Exit");
            System.exit(0);
            break;
        default:
            System.out.println("Invalid input");

}main();}
public static void main(String[] args) {
    details();
    main();
```

Output:



```
'nMENU DISPLAY
 1-show file in Ascending Order
 2-to view file operations
 3-to Exit from the application
ress the required function listed to carry out further steps
```

Enable initial menu options for the user to select the required data.

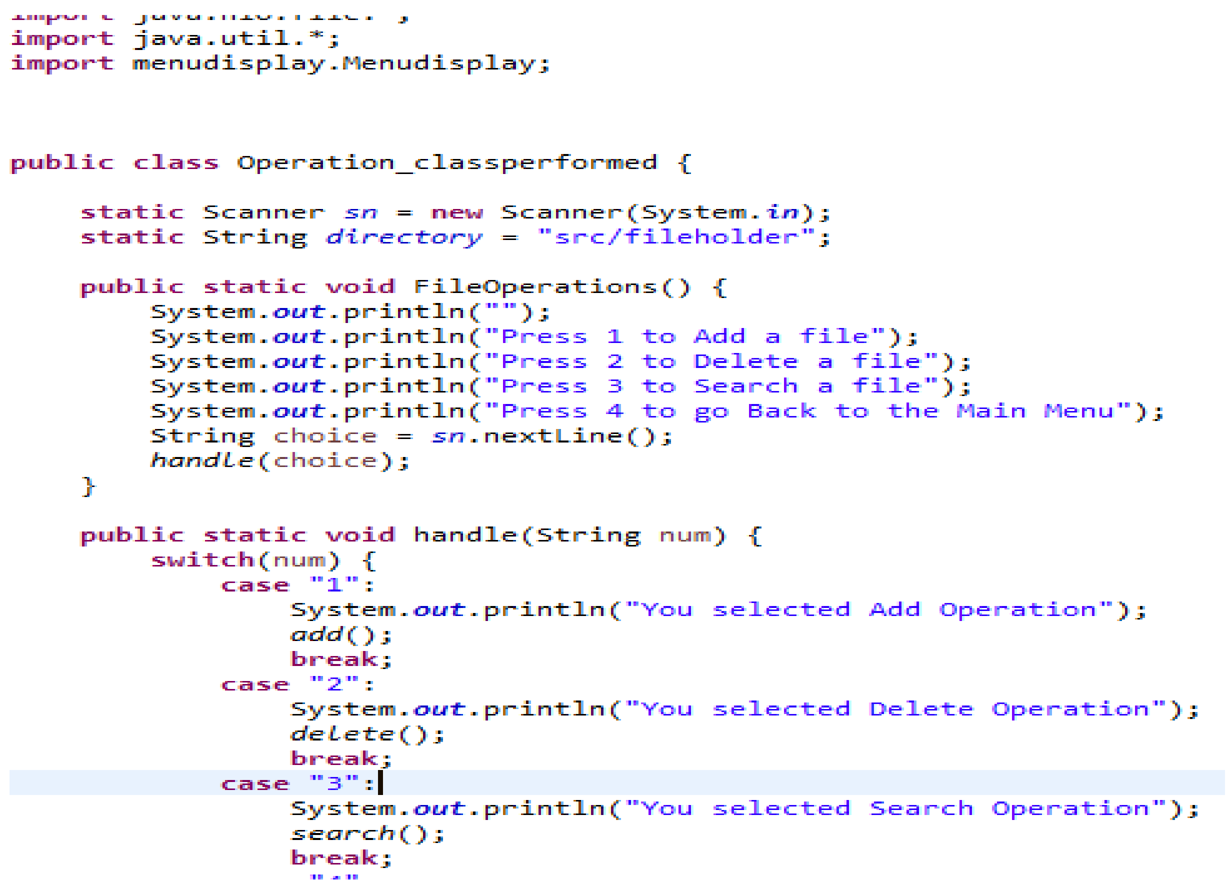
When a particular switch case is opted,

Case-1: the program sorts the file in ascending order and display the output

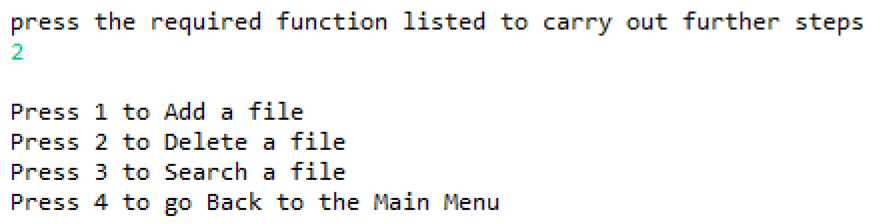
Case-2-the program takes the user to progress to secondary menu where the further operations are done.

Case-3: the console exits from the program and displays exit.

Provide the secondary operations and menu for user defined actions.



```
import java.util.*;
import menudisplay.Menudisplay;


public class Operation_classperformed {

    static Scanner sn = new Scanner(System.in);
    static String directory = "src/fileholder";

    public static void FileOperations() {
        System.out.println("");
        System.out.println("Press 1 to Add a file");
        System.out.println("Press 2 to Delete a file");
        System.out.println("Press 3 to Search a file");
        System.out.println("Press 4 to go Back to the Main Menu");
        String choice = sn.nextLine();
        handle(choice);
    }

    public static void handle(String num) {
        switch(num) {
            case "1":
                System.out.println("You selected Add Operation");
                add();
                break;
            case "2":
                System.out.println("You selected Delete Operation");
                delete();
                break;
            case "3":
                System.out.println("You selected Search Operation");
                search();
                break;
```



```
press the required function listed to carry out further steps
2

Press 1 to Add a file
Press 2 to Delete a file
Press 3 to Search a file
Press 4 to go Back to the Main Menu
```

The secondary menu helps us to enlist the operations and further state for the user to carry out.

CODE:

```
package menudisplay;

import java.util.*;

import ascending.Ascending_sorting;

import operation.Operation_classperformed;




public class Menudisplay {

        static Scanner sn=new Scanner(System.in);

 public static void details() {

         System.out.println("Menu display");

         System.out.println("vishnu");

         System.out.println("employee id:2527463");

         System.out.println(" accept the user input to select operations as per the requirement");}

 public static void main() {

         System.out.println("/nMENU DISPLAY");

         System.out.println(" 1-show file in Ascending Order");

                                         System.out.println(" 2-to view file operations");

                                         System.out.println(" 3-to Exit from the application");

                                         System.out.println("press the required function listed to carry out further steps");
```

```
                                        int choice = sn.nextInt();

                                        handle(choice);

}

public static void handle(int value) {

        switch(value)

        {case 1:

                 Ascending_sorting. ascendingvalues();

                        break;

                case 2:

                        Operation_classperformed .FileOperations();

                        break;

                case 3:

                        System.out.println("Exit");

                        System.exit(0);

                        break;

                default:

                        System.out.println("Invalid input");

}main();}

public static void main(String[] args) {

        details();

        main();}}
```

//CASE-1 SORTING

```
package ascending;

import java.io.*;

import java.util.*;
```

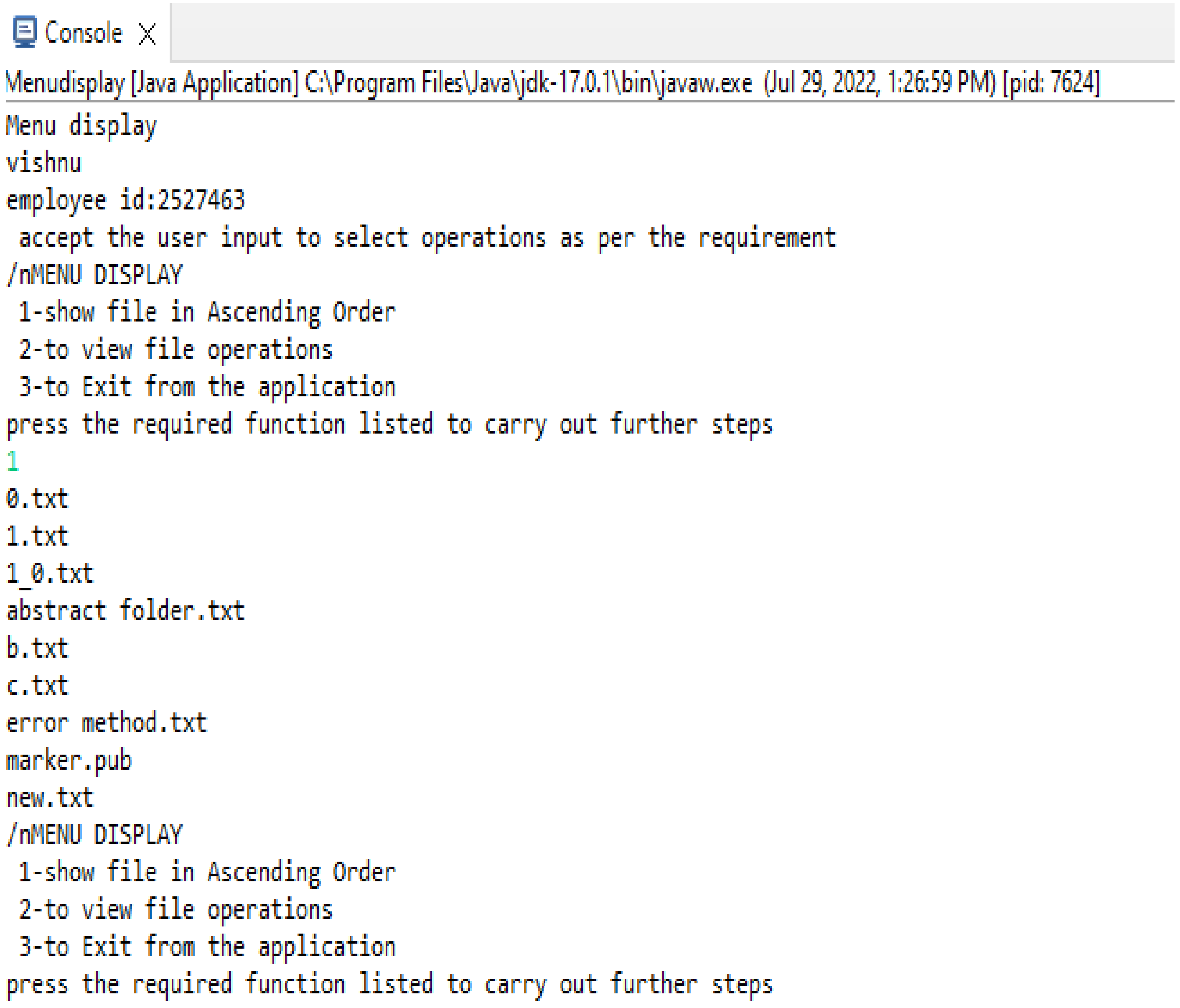
```
public class Ascending_sorting {

static String directory="src/fileholder";

        public static void ascendingvalues(){

                File[] files = new File(directory).listFiles();

                Set<String> a = new TreeSet<>();

                for(File file : files) {

                        if (!file.isFile()) {

                                continue;}

                        a.add(file.getName());}

                a.forEach(i->System.out.println(i));}}}
```

CASE-2: FILE OPERATIONS:

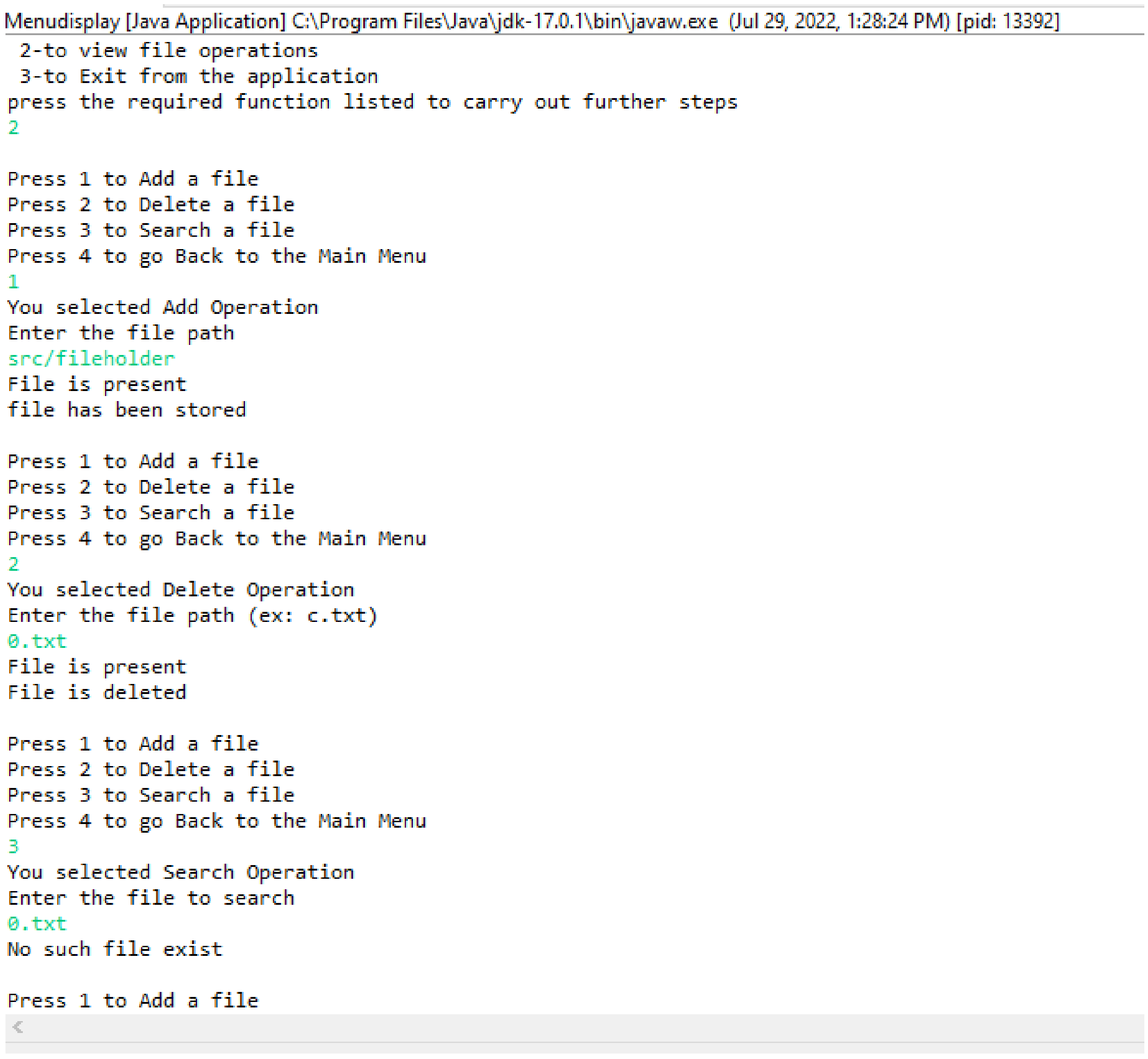
```
package operation;
import java.io.*;
import java.nio.file.*;
import java.util.*;
import menudisplay.Menudisplay;



public class Operation_classperformed {

	static Scanner sn = new Scanner(System.in);
	static String directory = "src/fileholder";

	public static void FileOperations() {
		System.out.println("");
		System.out.println("Press 1 to Add a file");
		System.out.println("Press 2 to Delete a file");
		System.out.println("Press 3 to Search a file");
		System.out.println("Press 4 to go Back to the Main Menu");
		String choice = sn.nextLine();
		handle(choice);
	}

	public static void handle(String num) {
		switch(num) {
			case "1":
				System.out.println("You selected Add Operation");
				add();
				break;
			case "2":
				System.out.println("You selected Delete Operation");
				delete();
				break;
			case "3":
				System.out.println("You selected Search Operation");
```

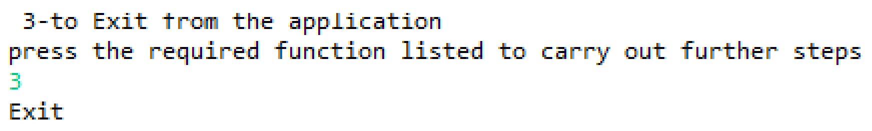
```java
                        search();
                        break;
                case "4":
                        System.out.println("Going Back to Main Menu");
                        Menudisplay.main();
                        break;
                default:
                        System.out.println("Invalid input");
                }
                FileOperations();
        }

        // to add a file

        public static void add() throws InvalidPathException {
                System.out.println("Enter the file path ");
                String input = sn.nextLine();
                Path path;
                try {
                        path = Paths.get(input);
                } catch (Exception e) {
                        System.out.println("Invalid input");
                        return;
                }

                if (!Files.exists(path)) {
                        System.out.println("No such file exist");
                        return;
                }else {
                        System.out.println("File is present");

                }

                String newPath = directory + "/" + path.getFileName();
                int i = 0;
                while (Files.exists(Paths.get(newPath))) {
                        i++;
                        newPath = directory + "/" + i + "_" + path.getFileName();
                }

                try {
                        Files.copy(path,  Paths.get(newPath));
                        System.out.println("file has been stored");
                } catch (IOException e) {
                        System.out.println("Not able to store the file");
                        System.out.println(e);
                }

        }

        // to delete a file

        public static void delete() throws InvalidPathException {
                System.out.println("Enter the file path (ex: c.txt)");
                String input = sn.nextLine();
                String Path = directory + "/" + input;
                Path path;

                try {
                        path = Paths.get(Path);
                } catch (Exception e) {
                        System.out.println("Invalid input");
                        return;
                }

                if (!Files.exists(path)) {
                        System.out.println("No such file existed,thus cannot be deleted");
                        return;
                } else {
                        System.out.println("File is present");
                }

                File Delete = new File(Path);
                try {
                        Delete.delete();
                        System.out.println("File is deleted");
                }
                catch (Exception e) {

                        System.out.println("Not able to delete file");
                        System.out.println(e);
                }
        }

        //to search a file

        public static void search() throws InvalidPathException{
                System.out.println("Enter the file to search ");
                String input = sn.nextLine();
                String Path = directory + "/" + input;
                Path path;

                try {
                        path = Paths.get(Path);
                } catch (Exception e) {
                        System.out.println("Invalid input");
                        return;
                }

                if(!Files.exists(path)) {
                        System.out.println("No such file exist");
                        return;
                } else {
                        System.out.println("File is present");
                }

        }
}
```

CONSOLE OUTPUTS:



```
Menudisplay [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Jul 29, 2022, 1:26:59 PM) [pid: 7624]
Menu display
vishnu
employee id:2527463
 accept the user input to select operations as per the requirement
/nMENU DISPLAY
 1-show file in Ascending Order
 2-to view file operations
 3-to Exit from the application
press the required function listed to carry out further steps
1
0.txt
1.txt
1_0.txt
abstract folder.txt
b.txt
c.txt
error method.txt
marker.pub
new.txt
/nMENU DISPLAY
 1-show file in Ascending Order
 2-to view file operations
 3-to Exit from the application
press the required function listed to carry out further steps
```


```
Menudisplay [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (Jul 29, 2022, 1:28:24 PM) [pid: 13392]
 2-to view file operations
 3-to Exit from the application
press the required function listed to carry out further steps
2

Press 1 to Add a file
Press 2 to Delete a file
Press 3 to Search a file
Press 4 to go Back to the Main Menu
1
You selected Add Operation
Enter the file path
src/fileholder
File is present
file has been stored

Press 1 to Add a file
Press 2 to Delete a file
Press 3 to Search a file
Press 4 to go Back to the Main Menu
2
You selected Delete Operation
Enter the file path (ex: c.txt)
0.txt
File is present
File is deleted

Press 1 to Add a file
Press 2 to Delete a file
Press 3 to Search a file
Press 4 to go Back to the Main Menu
3
You selected Search Operation
Enter the file to search
0.txt
No such file exist

Press 1 to Add a file
```


```
 3-to Exit from the application
press the required function listed to carry out further steps
3
Exit
```

CORE CONCEPTS USED:

- OOPS CONCEPT
- FILE HANDLING
- EXCEPTION HANDLING
- CONDITIONAL STATEMENTS

GITHUB LINK:

https://github.com/akash-n/assesment

UNIQUE POINTS OF THE PROGRAM:

1. The application has been designed to take continuous inputs even during exception states. The program does not stop due to error and throw able value.TO prevent the Stop the program ,we have opt for the correct option.

2. The application can take any file/folder name as input. Even if the user wants to create nested folder structure, user can specify the relative path, and the application takes care of creating the required folder structure.

3. User is also provided the option to write content if they want into the newly created file.

4. The application doesn't restrict user to specify the exact filename to search/delete file/folder. They can specify the starting input, and the program searches all files/folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

5. The application also allows user to delete folders which are not empty

CONCLUSION:

Thus, all the required data modulation according to the problem statement has been designed. Further, enhancement or modification is also available to

I. Append the file.
II. Retrieve the last modified data, file etc.
III. To ask for confirmation of a directory if the particular directory is not empty.
IV. To permit certain users only to access the files.