**DOCUMENTATION**

**Virtual Key for Your Repositories**

**Name :- Shivapriya K**

**CHAPTER**

**1. Sprint Planning And Task Completion**

**2. Core Concepts Used In The Project**

**3. Flow Of The Application**

**4. Demonstrate The Working of Application**

**5. Conclusion**

**1. Sprint Planning And Task Completion**

* + This project is planned in one Sprint
  + Creating the flow of the Application
  + Initialized the Git Repository
  + ([FirstProgram/FirstProject/src/myPack1 at master · Shivapriya29/FirstProgram (github.com)](https://github.com/Shivapriya29/FirstProgram/tree/master/FirstProject/src/myPack1))
  + Written the java program / Developed the Application to full fill all the requirements of the project definition
  + Testing the project on various inputs
  + Push on the Github
  + Creation of specification Document

**2. Core Concepts Used In The Project**

**Java File Handling :-**

The File class from the java.io package, allows us to work with files.

To use the File class, create an object of the class, and specify the filename or directory name

We Use Several Methods Using This File Class

1. createNewFile() :- Creates an empty file
2. delete() :- Deletes a file
3. getName() :- Returns the name of the file
4. list() :- Returns an array of the files in the directory
5. mkdir() :- Creates a directory

**Sorting :-**

Sorting refers to the operation or technique of arranging and rearranging sets of data in some specific order. A collection of records called a list where every record has one or more fields. The fields which contain a unique value for each record is termed as the key field. For example, a phone number directory can be thought of as a list where each record has three fields - 'name' of the person, 'address' of that person, and their 'phone numbers'. Being unique phone number can work as a key to locate any record in the list.

**Flow Control Statements :-**

In This Program We Use This Control Statements

* + **If-else**

**if**(f1.isDirectory()){

System.***out***.println("Folder is created successfully .\n\n");

}**else**{

System.***out***.println("Error Found ! \n\n");

}

* + **Switch Case**

**switch**(inputCase) {

**case** 1:

sort.Sorting(path);

**break**;

**case** 2:

System.***out***.println("File Operations \n");

FileObject obj = **new** FileObject();

terminateflag = obj.fileoperations(path);

**break**;

**case** 3: // Terminate The Applications .

**break**;

**default** :

System.***out***.println("Enter inputCase correctly and retry .... \n\n");

}

* + **do – while**

**do** {

System.***out***.println("Choose Option :\n\n 1. Retrive files from main Folder in asscending order \n 2. File Operations \n 3. Terminate the program sucessful\n");

inputCase = sc.nextInt();

**switch**(inputCase) {

**case** 1:

sort.Sorting(path);

**break**;

**case** 2:

System.***out***.println("File Operations \n");

FileObject obj = **new** FileObject();

terminateflag = obj.fileoperations(path);

**break**;

**case** 3: // Terminate The Applications .

**break**;

**default** :

System.***out***.println("Enter inputCase correctly and retry .... \n\n");

}

**if**(terminateflag == 1)

inputCase = 3;

} **while**(inputCase != 3);

## Exceptions:-

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

Java try and catch:-

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

**try {**

**if(file.createNewFile()) {**

**System.out.println("File created Sucessful .");**

**}**

**else**

**System.out.println("File created Unsucessful ");**

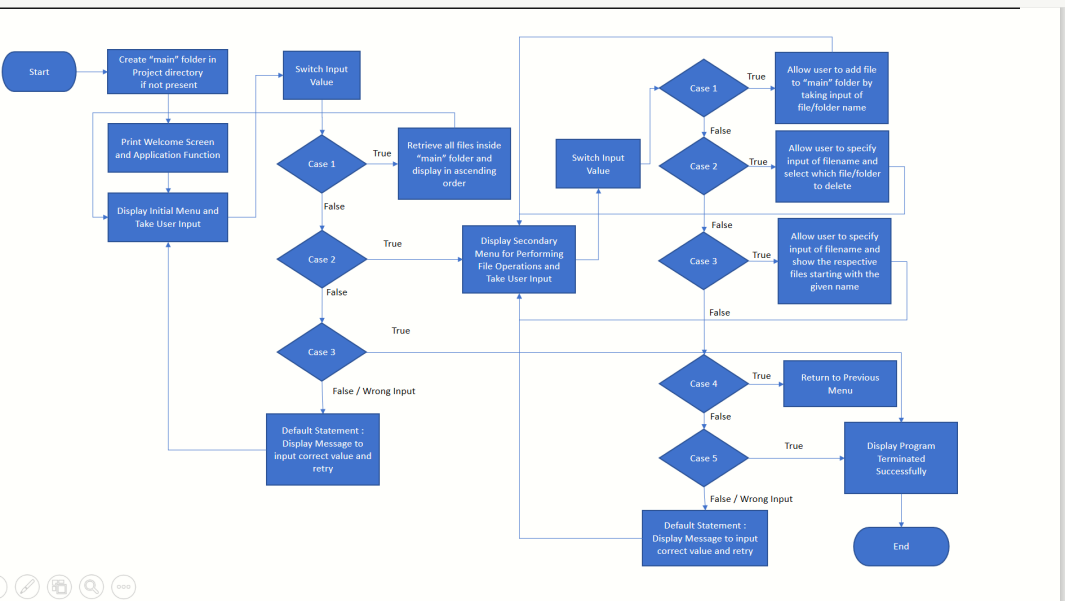
**} catch (IOException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**3. Flow Of The Application**

******

**4. Demonstrate The Working of Application**

* **Creation Of Floder:-**

**Program:-**

System.***out***.println("Enter the path where you want to create a folder: ");

Scanner sc = **new** Scanner(System.***in***);

String path = sc.next();

//Using Scanner class to get the folder name from the user

System.***out***.println("Enter the name of the main Directory ");

String folder = sc.next();;

path = path+folder;

File f1 = **new** File(path);

//Creating a folder using mkdir() method

**if** (!f1.exists()){

f1.mkdirs();

}

**if**(f1.isDirectory()){

System.***out***.println("Folder is created successfully .\n\n");

}**else**{

System.***out***.println("Error Found ! \n\n");

}

**OutPut:-**

Hello SIMPLILEARN

WellCome To MY Program.

SHIVAPRIYA HERE.

Enter the path where you want to create a folder:

S:\git\newRrpHttp\myRep

Enter the name of the main Directory

Floder 1

Folder is created successfully .

* **Welcome to the Application Functionality:-**

**Program:-**

System.***out***.println("Welcome to the Application Fuctionality . \n");

**int** inputCase;

Sorting sort = **new** Sorting();

**int** terminateflag = 0;

**do** {

System.***out***.println("Choose Option :\n\n 1. Retrieve files from main Folder in ascending order \n 2. File Operations \n 3. Terminate the program successful\n");

inputCase = sc.nextInt();

**switch**(inputCase) {

**case** 1:

sort.Sorting(path);

**break**;

**case** 2:

System.***out***.println("File Operations \n");

FileObject obj = **new** FileObject();

terminateflag = obj.fileoperations(path);

**break**;

**case** 3: // Terminate The Applications .

**break**;

**default** :

System.***out***.println("Enter inputCase correctly and retry .... \n\n");

}

**Output:-**

Welcome to the Application Fuctionality .

Choose Option :

1. Retrive files from main Folder in asscending order

2. File Operations

3. Terminate the program sucessful

Total number of items present in the directory: 0

Choose Option :

1. Retrive files from main Folder in asscending order

2. File Operations

3. Terminate the program sucessful

1

Total number of items present in the directory: 0

* **File Operations:-**

**Input:-**

**do** {

System.***out***.println("Choose Option :\n \n 1.Add file\n 2.Delete file\n 3.Display Files start with same name . \n 4. Return to previous menu :\n 5. Terminate the Application \n");

inputCase = sc.nextInt();

**switch**(inputCase) {

**case** 1: //Creation of File in the given folder.

System.***out***.println("Enter filename to create :");

fileName = sc.next();

file = **new** File(path, fileName);

**try** {

**if**(file.createNewFile()) {

System.***out***.println("File created Sucessful .");

}

**else**

System.***out***.println("File created Unsucessful ");

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**break**;

**case** 2: // Deletion of File from the given folder.

System.***out***.println("Enter filename to delete :");

fileName = sc.next();

file = **new** File(path, fileName);

**if**(file.delete())

{

System.***out***.println("File deleted successfully"+ file.getName());

}

**else**

{

System.***out***.println("Failed to delete the file");

}

**break**;

**case** 3: // Display the Files with given name in the given folder.

System.***out***.println("Enter filename to Display :");

String filname = sc.next();

File folder = **new** File(path);

File[] filelist = folder.listFiles();

ArrayList <String> list = **new** ArrayList<String>();

**for**(File f:filelist) {

list.add(f.getName());

}

System.***out***.println("Files with fileName start with :" + filname );

**for**(String l:list) {

**if**(l.contains(filname))

System.***out***.println(l);

}

**break**;

**case** 4: // Return to Previous Menu

inputCase = 5;

**break**;

**case** 5: // Terminate the application .

**return** 1;

**default** :

System.***out***.println(" Enter correct inputCase and retry ..... \n");

**break**;

}

}**while**(inputCase != 5) ;

**Output:-**

File Operations

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

1

Enter filename to create :

s

File created Sucessful .

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

1

Enter filename to create :

d

File created Sucessful .

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

1

Enter filename to create :

f

File created Sucessful .

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

4

Choose Option :

1. Retrieve files from main Folder in ascending order

2. File Operations

3. Terminate the program sucessful

1

Total number of items present in the directory: 3

d

f

s

Choose Option :

1. Retrieve files from main Folder in ascending order

2. File Operations

3. Terminate the program sucessful

2

File Operations

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

1

Enter filename to create :

g

File created Sucessful .

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

4

Choose Option :

1. Retrieve files from main Folder in ascending order

2. File Operations

3. Terminate the program sucessful

1

Total number of items present in the directory: 4

d

f

g

s

Choose Option :

1. Retrive files from main Folder in asscending order

2. File Operations

3. Terminate the program sucessful

2

File Operations

Choose Option :

1.Add file

2.Delete file

3.Display Files start with same name .

4. Return to previous menu :

5. Terminate the Application

2

Enter filename to delete :

s

File deleted successfullys

* **Retrive files from main Folder in asscending order**

**Input:-**

File folder = **new** File(dirPath);

**if**(folder.isDirectory())

{

File[] fileList = folder.listFiles();

Arrays.*sort*(fileList);

System.***out***.println("\nTotal number of items present in the directory: " + fileList.length );

// Lists only files since we have applied file filter

**for**(File file:fileList)

{

System.***out***.println(file.getName());

}

**Output:-**

Choose Option :

1. Retrive files from main Folder in asscending order

2. File Operations

3. Terminate the program sucessful

1

Total number of items present in the directory: 3

d

f

g

* **Terminating Program**

**Input:-**

**case** 5: // Terminate the application .

**return** 1;

**default** :

System.***out***.println(" Enter correct inputCase and retry ..... \n");

**break**;

}

}**while**(inputCase != 5) ;

**return** 0;

}

**Output:-**

Choose Option :

1. Retrieve files from main Folder in ascending order

2. File Operations

3. Terminate the program successful

3

Application sucessfully Terminated !!

**5. Conclusion:-** In The project File Operations like creating file ,deleting file ,searching file, break the file ,are used Retrieve files from main Folder in ascending order,using sorting and Exception handling (try cathch) operations are used.