**LockedMe.com**

* This document contains sections for:
* [Sprint planning and Task completion](#Sprint_plan)
* [Core concepts used in project](#Core_concepts)
* [Flow of the Application](#Flow).
* [Demonstrating the product capabilities, appearance, and user interactions.](#Product_capability)
* [Unique Selling Points of the Application](#USP)
* [Conclusions](#Conclusions)

The project is developed by **Aditya Kumar Gupta.**

The code for this project is hosted at <https://github.com/Aditya-dev-ind/Lockedme.com>.

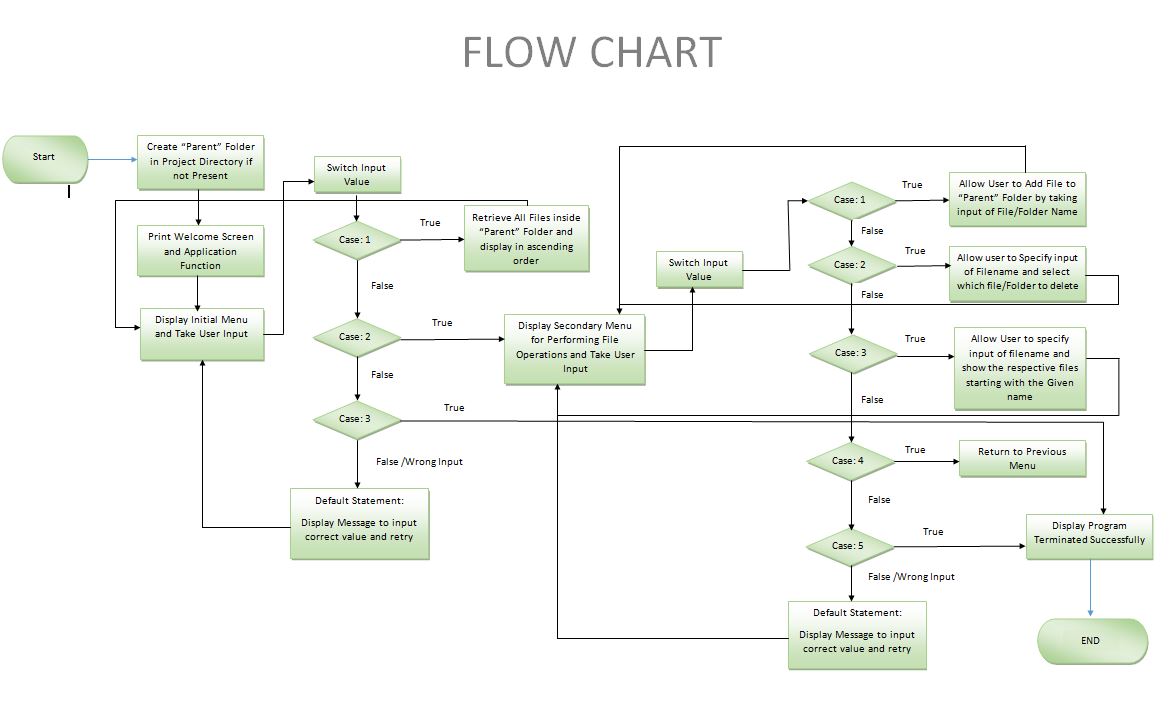
## Sprints planning and Task completion

* The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:
* Creating the flow of the application
* Initializing git repository to track changes as development progresses.
* Writing the Java program to fulfill the requirements of the project.
* Testing the Java program with different kinds of User input
* Pushing code to GitHub.
* Creating this specification document highlighting application capabilities, appearance, and user interactions.

## Core concepts used in project

* Array, File Handling, Sorting, Flow Control, Exception Handling, Loops.

## Flow of the Application



## Demonstrating the product capabilities, appearance, and user interactions

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1. [Creating the project in Eclipse](#Step_1)
2. [Writing a program in Java for the application(**FileManager.java**)](#Step_2)
3. Created a Library class (**File\_Handling**) for core operations through different methods.
4. [Pushing the code to GitHub repository](#Step_6)

## **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 **FileManager** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

## **Step 2:** Writing a program in Java for the entry point of the application (**FileManager.java**)

**public** **class** FileManager

{

//Main Method

**public** **static** **void** main(String[] args)

{

File\_Handling fh = **new** File\_Handling();

//Called this function from Library class to check if directory does not exist then create one.

fh.MainDirectory();

fh.Welcome() ;

}

}

## **Step 3:** Writing a program in Java to display Menu options available for the user (**FileManager.java**)

* **Library Class File\_Handling** consists methods for -:
  1. [Displaying Welcome Screen](#Step_3_1)
  2. [Displaying Initial Menu](#Step_3_2)
  3. [Displaying Secondary Menu for File Operations available](#Step_3_3)

**Step 3.1:** Writing method to display Welcome Screen

**void** Welcome()

{

System.***out***.println("»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»");

System.***out***.println("-> Welcome to LockedMe.com");

System.***out***.println("-> Developed by:Aditya Kumar Gupta.\n");

System.***out***.println(" <<----------------Application Highlights-------------------->>");

System.***out***.println("-> This Application is used for handling file operations i.e Creation & Deletion of Folder/File alongwith Searching.");

System.***out***.println("-> Press (Enter key) to continue....");

System.***out***.println("»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»");

**try**

{

System.***in***.read();

DisplayInitial();

}

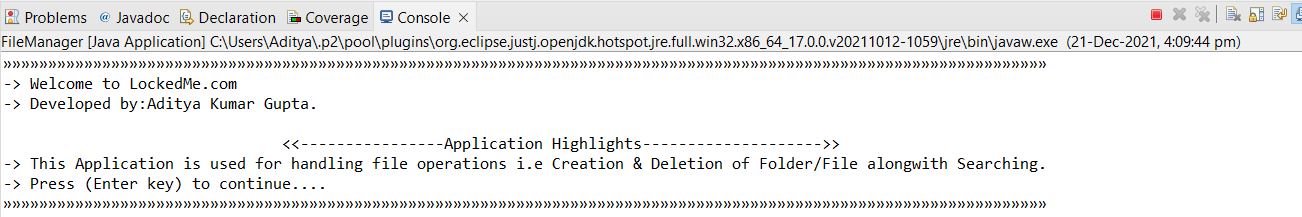
**catch**(Exception e)

{ e.printStackTrace();

}

}

**Output:**



**Step 3.2:** Writing method to display Initial Menu

**void** DisplayInitial()

{

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

**try**

{

**int** i=5;

**do**

{

System.***out***.println("Press 1 to Retrive Folder 'Parent' & Display the contents inside it(in asc.order).");

System.***out***.println("Press 2 to Perform File Operations.");

System.***out***.println("Press 3 to Terminate application.");

choose1 = opr.nextInt();

**if**(choose1==1)

{

RetrieveDir();

DisplayInitial();

**break**;

}

**else** **if**(choose1==2)

{

Operations();

**break**;

}

**else** **if**(choose1==3)

{

terminate();

**break**;

}

**else**

{

System.***out***.println("Please choose correct option to Continue!");

}

}

**while**(i<6);

}

**catch**(Exception e)

{

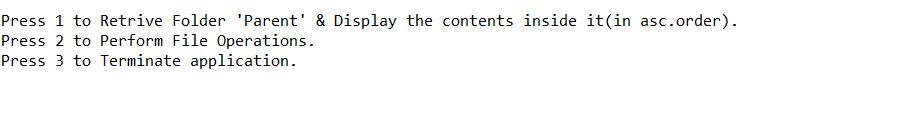
DisplayInitial();

}

opr.close();

}

**Output:**



**Step 3.3:** Writing method to display Secondary Menu for File Operations

**void** Operations()

{

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

**try**

{

**int** i=5;

**do**

{

System.***out***.println("Press 1 For Creation of New Folder/File in 'Parent' Directory.");

System.***out***.println("Press 2 For Deleting of Existing Folder/File in 'Parent' Directory.");

System.***out***.println("Press 3 For Searching File in 'Parent' Directory.");

System.***out***.println("Press 4 To Return to Previous Menu.");

System.***out***.println("Press 5 to Terminate application. ");

choose = opr.nextInt();

**if**(choose==1)

{

creating\_File();

**break**;

}

**else** **if**(choose==2)

{

Deleting\_File();

**break**;

}

**else** **if**(choose==3)

{

Searchtxt();

**break**;

}

**else** **if**(choose==4)

{

DisplayInitial() ;

**break**;

}

**else** **if**(choose==5)

{

terminate();

**break**;

}

**else**

{

System.***out***.println("Please choose correct option to Continue!");

}

}

**while**(i<6);

}

**catch** (Exception e)

{

System.***out***.println("Please choose correct option to Continue!");

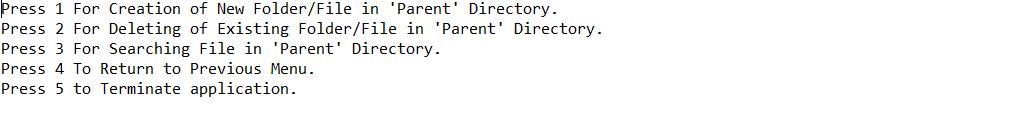
Operations() ;

}

opr.close();

}

**Output:**



## **Step 4:** Writing a program in Java to handle Menu options selected by user (**FileManager.java**)

* DisplayInitial() & Operations() methods used for -:
  1. [Handling input selected by user in initial Menu](#Step_4_1)
  2. [Handling input selected by user in secondary Menu for File Operations](#Step_4_2)

**Step 4.1:** Writing method to handle user input in initial Menu

**void** DisplayInitial()

{

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

**try**

{

**int** i=5;

**do**

{

System.***out***.println("Press 1 to Retrive Folder 'Parent' & Display the contents inside it(in asc.order).");

System.***out***.println("Press 2 to Perform File Operations.");

System.***out***.println("Press 3 to Terminate application.");

choose1 = opr.nextInt();

**if**(choose1==1)

{

RetrieveDir();

DisplayInitial();

**break**;

}

**else** **if**(choose1==2)

{

Operations();

**break**;

}

**else** **if**(choose1==3)

{

terminate();

**break**;

}

**else**

{

System.***out***.println("Please choose correct option to Continue!");

}

}

**while**(i<6);

}

**catch**(Exception e)

{

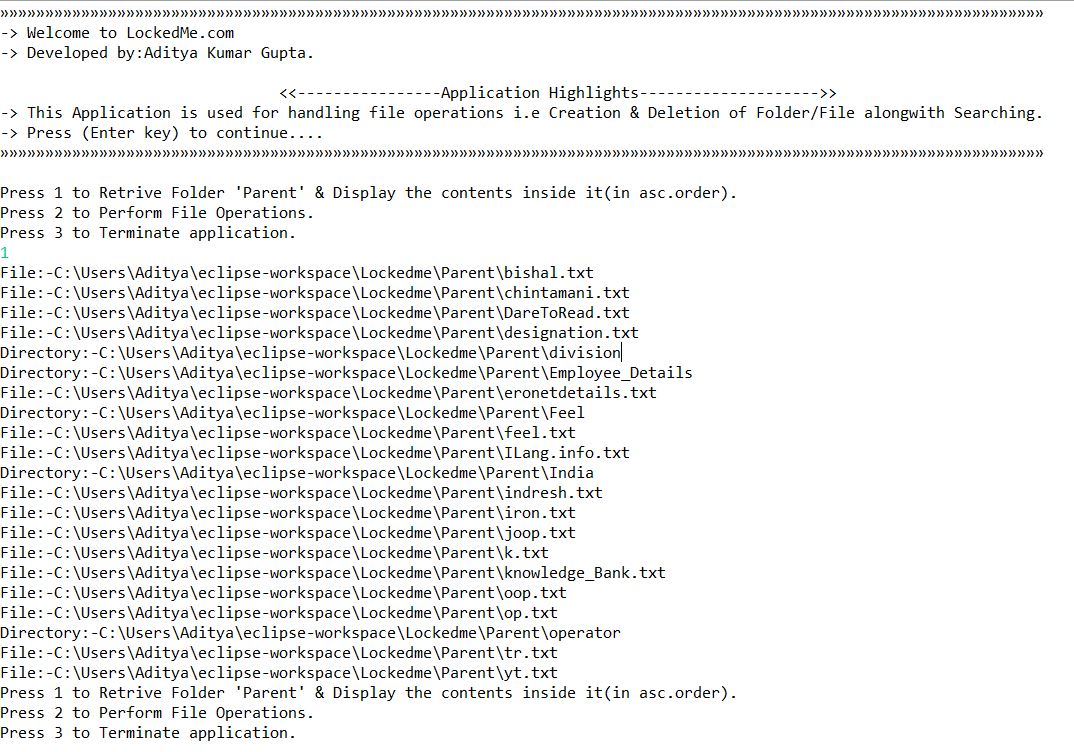
DisplayInitial();

}

opr.close();

}

**Output:**



**Step 4.2:** Writing method to handle user input in Secondary Menu for File Operations

**void** Operations()

{

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

**try**

{

**int** i=5;

**do**

{

System.***out***.println("Press 1 For Creation of New Folder/File in 'Parent' Directory.");

System.***out***.println("Press 2 For Deleting of Existing Folder/File in 'Parent' Directory.");

System.***out***.println("Press 3 For Searching File in 'Parent' Directory.");

System.***out***.println("Press 4 To Return to Previous Menu.");

System.***out***.println("Press 5 to Terminate application. ");

choose = opr.nextInt();

**if**(choose==1)

{

creating\_File();

**break**;

}

**else** **if**(choose==2)

{

Deleting\_File();

**break**;

}

**else** **if**(choose==3)

{

Searchtxt();

**break**;

}

**else** **if**(choose==4)

{

DisplayInitial() ;

**break**;

}

**else** **if**(choose==5)

{

terminate();

**break**;

}

**else**

{

System.***out***.println("Please choose correct option to Continue!");

}

}

**while**(i<6);

}

**catch** (Exception e)

{

System.***out***.println("Please choose correct option to Continue!");

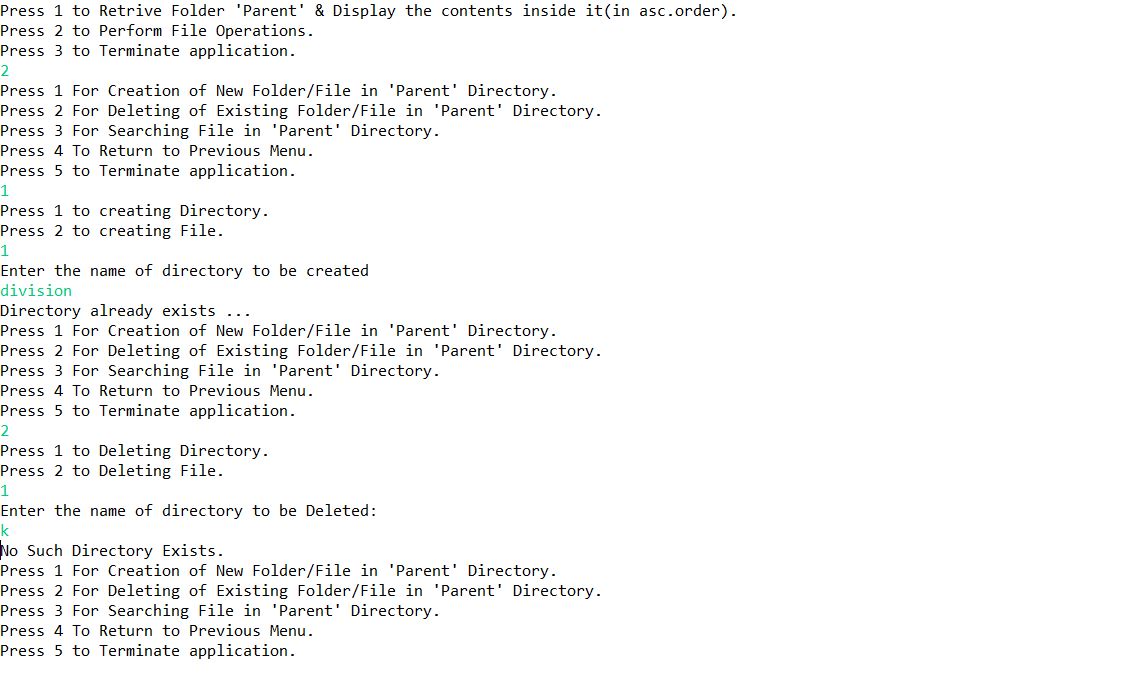
Operations() ;

}

opr.close();

}

**Output:**



## **Step 5:** Writing a program in Java to perform the File operations as specified by user (**FileManager.java**)

* **Library Class File\_Handling** consists methods for -:
  1. [Creating “Parent” folder in project if it’s not already present](#Step_5_1)
  2. [Displaying all files in “Parent” folder in ascending order and also with directory structure.](#Step_5_2)
  3. [Creating a file/folder as specified by user input.](#Step_5_3)
  4. [Search files as specified by user input in “Parent” folder and it’s subfolders.](#Step_5_4)
  5. [Deleting a file/folder from “Parent” folder](#Step_5_5)

**Step 5.1:** Writing method to create “Parent” folder in project if it’s not present

**void** Main Directory()

{

**boolean** success = **false**;

String dir = "Parent";

//Creating Directory At the beginning of Application

File directory = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\"+dir);

**if** (directory.exists())

{ //No Message

} **else**

{

// No Message

success = directory.mkdir();

**if** (success)

{

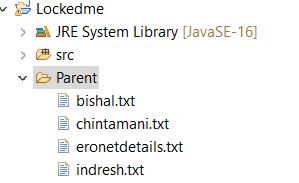
//No Message

}

}

}

**Output:**



**Step 5.2:** Writing method to display all files in “Parent” folder in ascending order and also with directory structure.

**void** RetrieveDir()

{

File dir = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\");

File array[]=dir.listFiles();

**if**(array.length>0)

{

**for** (**int** i=0;i<=array.length;i++)

{

**if**(array[i].isFile())

{

System.***out***.println("File:-"+array[i]);

}

**else** **if**(array[i].isDirectory())

{

System.***out***.println("Directory:-"+array[i]);

}

}

}

**else**

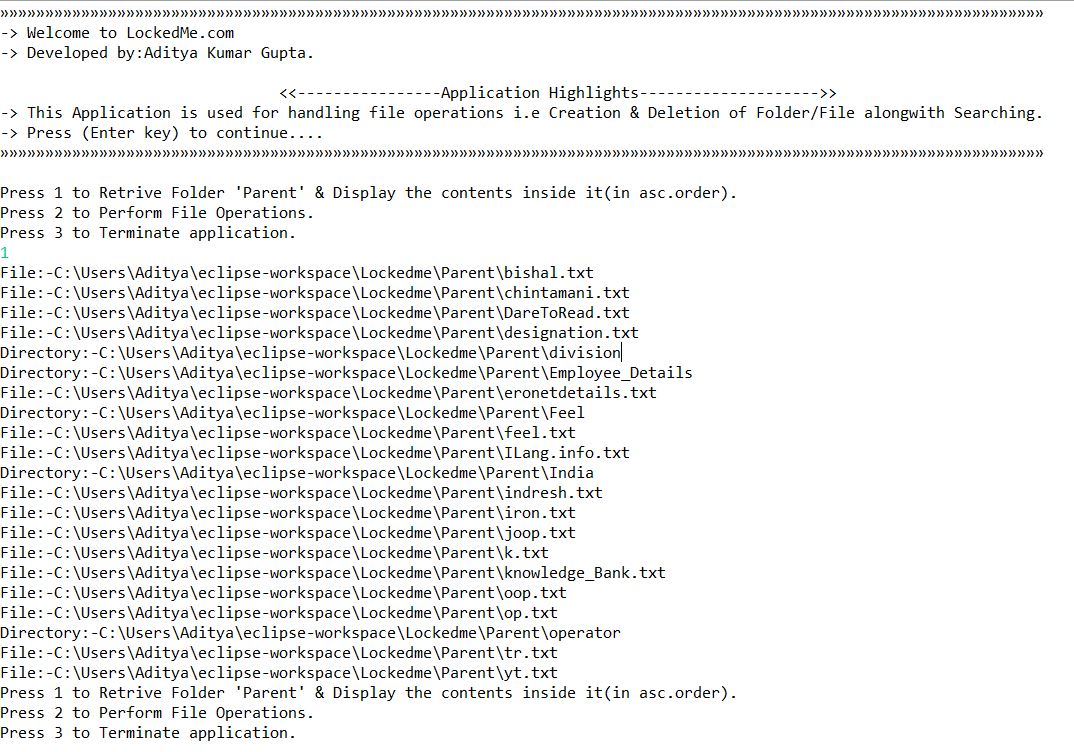
{

System.***out***.println("There is no files in Directory.");

}

}

**Output:**



**Step 5.3:** Writing method to create a file/folder as specified by user input.

**void** creating\_File()

{

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

**int** i=5;

**do**

{

System.***out***.println("Press 1 to creating Directory.");

System.***out***.println("Press 2 to creating File.");

option = opr.nextInt();

**if**(option==1)

{

**boolean** success = **false**;

//Taking User input For creation of Directory

System.***out***.println("Enter the name of directory to be created");

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

String dir = reader.nextLine();

// Checking whether directory created or not

File directory = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\"+dir);

**if** (directory.exists())

{

System.***out***.println("Directory already exists ...");

Operations() ;

} **else**

{

System.***out***.println("Directory not exists, creating now");

success = directory.mkdir();

**if** (success)

{

System.***out***.printf("Successfully created new directory : %s%n", dir);

Operations() ;

}

**else**

{

System.***out***.printf("Failed to create new directory: %s%n", dir);

Operations() ;

}

}

reader.close();

}

**else** **if**(option==2)

{

**boolean** success = **false**;

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

// For Creating New File

System.***out***.println("Enter file name to be created. ");

String filename = reader.nextLine();

// Checking whether File created or not

File f = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\"+filename+".txt");

**if** (f.exists() )

{

System.***out***.println("File already exists.");

Operations() ;

}

**else**

{

System.***out***.println("No such file exists, creating now..");

**try**

{

success = f.createNewFile();

}

**catch** (IOException e)

{

e.printStackTrace();

}

**if** (success)

{

System.***out***.printf("Successfully created new file: %s%n", f);

Operations() ;

}

**else**

{

System.***out***.printf("Failed to create new file: %s%n", f);

Operations() ;

}

}

reader.close();

}

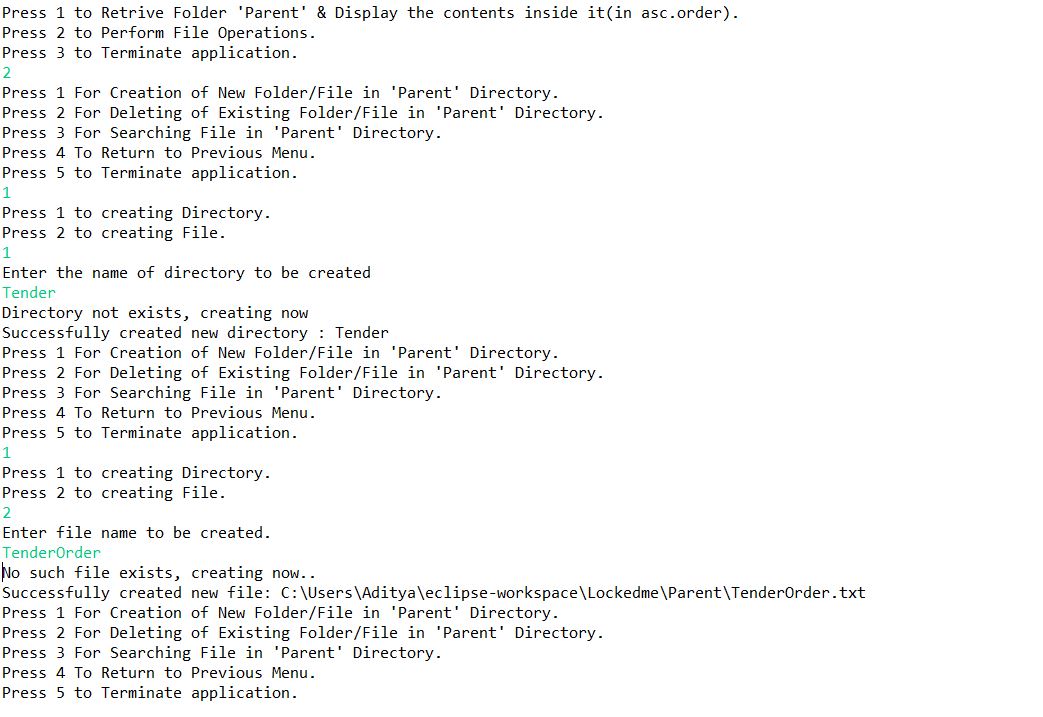
}

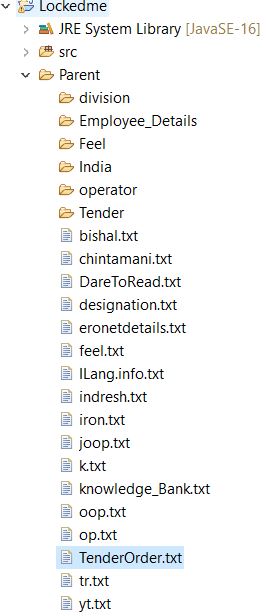
**while**(i<6);

opr.close();

}

**Output:**





**Step 5.4:**  Writing method to search for all files as specified by user input in “Parent” folder.

**void** Searchtxt()

{

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

File dir = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\");

File array[]=dir.listFiles();

System.***out***.println("Please Enter File/Folder name, which you want to search in 'Parent'!");

SearchFile=input.next();

String txt= SearchFile;

**try**

{

**for** (**int** i=0;i<=array.length;i++)

{

**if**(array[i].getName().startsWith(txt))

{

System.***out***.println("File/Folder Found:- " +array[i]);

}

}

}

**catch** (Exception e)

{

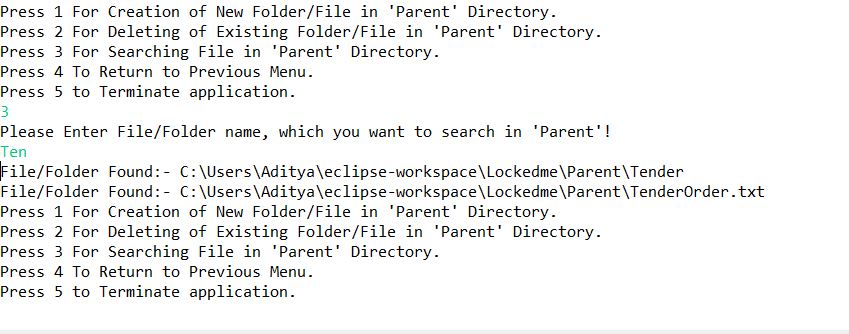
Operations();

}

input.close();

}

**Output:**



**Step 5.5:**  Writing method to delete file/folder specified by user input in “Parent”.

**void** Deleting\_File()

{

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

**int** i=5;

**do**

{

System.***out***.println("Press 1 to Deleting Directory.");

System.***out***.println("Press 2 to Deleting File.");

option = opr.nextInt();

**if**(option==1)

{

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

**boolean** success = **false**;

// User Input For Directory Deleting

System.***out***.println("Enter the name of directory to be Deleted:");

String dir = reader.nextLine();

File directory = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\"+dir);

//Checking whether directory Present or not

**if** (directory.exists())

{

success = directory.delete();

**if** (success)

{

System.***out***.printf("Successfully Deleted directory : %s%n", dir);

Operations() ;

}

**else**

{

System.***out***.printf("Failed to delete directory,First Delete the file inside Directory then delete the folder.: %s%n", dir);

Operations() ;

}

}

**else**

{

System.***out***.println("No Such Directory Exists.");

Operations() ;

}

reader.close();

}

**else** **if**(option==2)

{

**boolean** success = **false**;

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

// User Input For File Deleting

System.***out***.println("Enter file name to be Deleted:");

String filename = reader.nextLine();

//Checking whether File Present or not

File f = **new** File("C:\\Users\\Aditya\\eclipse-workspace\\Lockedme\\Parent\\"+filename+".txt");

**if** (f.exists() )

{

success = f.delete();

**if** (success )

{

System.***out***.printf("Successfully deleted file: %s%n", f);

Operations() ;

}

**else**

{

System.***out***.printf("Failed to delete file: %s%n", f);

Operations() ;

}

}

**else**

{

System.***out***.println("No such File exists!");

Operations() ;

}

reader.close();

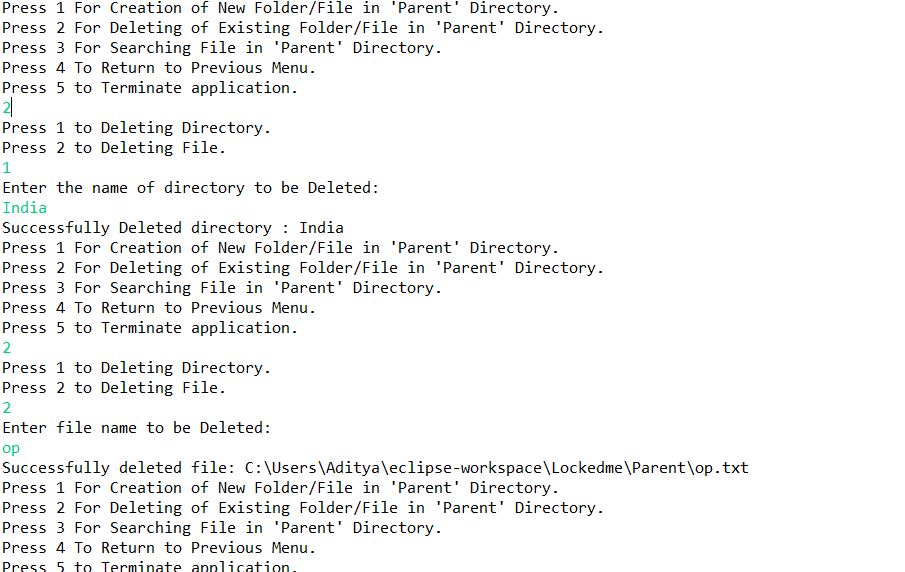
}

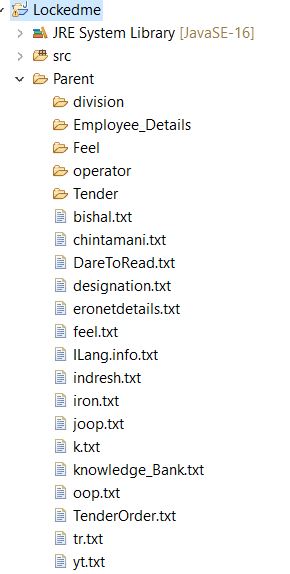
}

**while**(i<6);

opr.close();

}

**Output:** 



## **Step 6:** Pushing the code to GitHub repository

1. Open your command prompt and navigate to the folder where you have created your files.
   1. **cd <folder path>**
2. Initialize repository using the following command:
   1. **git init**
3. Add all the files to your git repository using the following command:
   1. **git add .**
4. Commit the changes using the following command:
   1. **git commit . -m <commit message>**
5. Push the files to the folder you initially created using the following command:
6. **git push -u origin master**

## 

## Unique Selling Points of the Application

1. The application is designed to keep on running and taking user inputs even after exceptions occur. To terminate the application, appropriate option needs to be selected.
2. The application can take any file/folder name as input and the application takes care of creating the required folder structure.
3. 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.
4. The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

## Conclusions

Further enhancements to the application can be made which may include:

* Allowing user to write data to the file.
* Allowing user to append data into the file.
* Allowing user to manually update the path where to save file/folder.
* Conditions to check if user is allowed to delete the file or add the file at the specific locations.
* Asking user to verify if they really want to delete the selected directory if it’s not empty.
* Retrieving files/folders by different criteria like Last Modified, Type, etc.