**Specification Document**

* **Project Name:** LockedMe.com
* **Developer Details:** Atreyee Mondal
* **Project Description:** The goal of this application is to develop a prototype, which will be presented to the relevant stakeholders for the budget approval by company Lockers Pvt. Ltd. Generic features of this application (implemented using core java) are as follows:
  + Retrieving the file names in an ascending order
  + Business-level operations:
    - Option to add a user specified file to the application
    - Option to delete a user specified file from the application
    - Option to search a user specified file from the application
    - Navigation option to close the current execution context and return to the main context
      * Option to close the application

This application is planned in 3 sprints and implemented using java then pushed to GitHub repository to track the enhancement of the application. The next sections describe about each step of the application.

* **Task achieved in each Sprint:**

Sprint 1:

* + 1. Develop command line UI for user interaction
    2. Develop Welcome Screen
    3. Develop Main Menu
    4. Develop Sub Menu
    5. Develop a feature to add a file
    6. Develop a feature to delete a user specified file

Sprint 2:

* + - * 1. Develop a feature to search a user specified file
        2. Develop a feature to show all the files in ascending order inside a directory
        3. Develop feature to navigate back to main menu
        4. Develop a feature to exit the application

Sprint 3:

1. Code Optimisation
2. Develop a feature which helps not to close the program until user choose
3. Develop exception handling if occurs in the program
4. Use Case Sensitivity to ensure correct file is deleted/ searched.

* **Flow Chart of the Application:**

View all Files in Ascending Order

Show Main Menu

Show Welcome Screen

Show Submenu

1

2

3

A

B

Add Files

Search Files

Delete Files

C

* **Core Concepts Used:**

Exception Handling, loops, collections, file handling, switch case, if-else, data structure, linear search, recursion.

* **Source Code:**

To implement the mentioned application, we have to create two java classes inside a package. The classes are namely Test and FileOperations to implement main method and specific file operations respectively. The following steps are done for creating a class inside com.lockers package.

1. Open Eclipse
2. *[Right click]* on the **src** folder of the project. (I named the project as LockedMe.com.)
3. Select *New* -> *Java Class* -> Enter the filename

Execute the code below.

1. **FileOperations.java**

**package** com.lockers;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.nio.charset.StandardCharsets;

**import** java.nio.file.Files;

**import** java.nio.file.Paths;

**import** java.util.Collections;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** FileOperations {

//This function will show all the files in ascending order.

**public** **static** **void** RecursivePrint(File[] arr,**int** index,**int** level)

{

// terminate condition

**if**(index == arr.length)

**return**;

// tabs for internal levels

**for** (**int** i = 0; i < level; i++)

System.***out***.print("\t");

// for files

**if**(arr[index].isFile())

System.***out***.println("File:" +arr[index].getName());

// for sub-directories

**else** **if**(arr[index].isDirectory())

{

System.***out***.println("Subdirectory: [" + arr[index].getName() + "]");

// recursion for sub-directories

*RecursivePrint*(arr[index].listFiles(), 0, level + 1);

}

// recursion for main directory

*RecursivePrint*(arr,++index, level);

}

//add user specified files to the existing directory

**public** **static** **void** createFileUsingFileClass(String fileName) **throws** IOException

{

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

//Create the file

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

System.***out***.println("File is added!");

//Write Content from UI

FileWriter writer = **new** FileWriter(file);

System.***out***.println("Type Yes to write content to the file:");

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

String response=s.next();

**if**(response.equals("Yes")||response.equals("yes")||response.equals("YES"))

{

System.***out***.println("Enter File Content:");

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

writer.write(sc.nextLine());

writer.close();

}

**else**

{

System.***out***.println("File content not written.");

}

}

**else**{

//if the file name already exist in the directory,the application will show it.

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

}

}

//this function is used to read file contents

**public** **static** List<String> readFileInList(String fileName)

{

List<String> lines = Collections.*emptyList*();

**try**

{

lines = Files.*readAllLines*(Paths.*get*(fileName), StandardCharsets.***UTF\_8***);

}

**catch** (IOException e)

{

System.***err***.println(e.getMessage());

}

**return** lines;

}

//this function is used to delete the user specified file from the existing directory.

**public** **static** **void** delete(String filename)

{

**boolean** a;

**try** {

a = Files.*deleteIfExists*(Paths.*get*(filename));

//if file found in the directory,it will get deleted.

**if** (a == **true**)

{

System.***out***.println(filename + "-Deleted Successfully.");

}

//if the user specified file is not in the directory it will show file not found.

**else**

{

System.***out***.println("File not Found");

}

} **catch** (IOException e) {

System.***err***.println(e.getMessage());

}

}

//This function will return the file if found while searching.

**public** **static** **void** search(String filename) {

File folder\_name=**new** File (System.*getProperty*("user.dir"));

String[] list = folder\_name.list();

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

**if** (filename.equals(file)) {

System.***out***.println("File Found : File " + filename + " exists at " + folder\_name);

**return**;

}

}

System.***out***.println("File not Found");

}

}

1. **Test.java (contains main method)**

**package com.lockers;**

**import java.io.File;**

**import java.io.IOException;**

**import java.util.Iterator;**

**import java.util.List;**

**import java.util.Scanner;**

**public class Test {**

**void showMainMenu() {**

**System.out.println("--------Main Menu--------"); //showing the main menu**

**System.out.println("1. View the list of files inside current directory \n2. View UI options \n3. Exit from Application");**

**System.out.println("Enter your choice:");**

**try{**

**Scanner s1 = new Scanner(System.in);**

**int option = s1.nextInt();**

**switch (option){**

**case 1 : {**

**String maindirpath = System.getProperty("user.dir");**

**// File object**

**File maindir = new File(maindirpath);**

**if(maindir.exists() && maindir.isDirectory())**

**{**

**// array for files and sub-directories**

**// of directory pointed by main\_directory**

**File arr[] = maindir.listFiles();**

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

**System.out.println("All Files from current working directory (in ascending order) : " + maindir);**

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

**// Calling recursive method**

**FileOperations.RecursivePrint(arr,0,0);**

**}**

**showMainMenu();**

**}**

**case 2 : {**

**showSubMenu();**

**}**

**case 3 : {**

**System.out.println("Thank You. End of application.");**

**System.exit(0);**

**}**

**default:**

**System.out.println("Invalid Input. Showing main menu...");**

**showMainMenu();**

**}**

**}**

**catch (Exception e){**

**System.out.println("Please enter 1, 2 or 3");**

**showMainMenu();**

**}**

**}**

**void showSubMenu() {**

**System.out.println("A. Add a file \nB. Delete a specific file \nC. Search for a specific file \nD. Main Memu ");**

**System.out.println("Enter your choice:");**

**Scanner s2 = new Scanner(System.in);**

**String c= s2.next();**

**switch (c){**

**case "A" :**

**{**

**Scanner s3=new Scanner(System.in);**

**System.out.println("Enter File Name:");**

**String filename=s3.next();**

**try {**

**FileOperations.createFileUsingFileClass(filename);**

**} catch (IOException e) {**

**System.err.println(e.getMessage());**

**}**

**List l = FileOperations.readFileInList(filename);**

**Iterator<String> itr = l.iterator();**

**while (itr.hasNext())**

**{**

**System.out.println("The content of the file is:"+itr.next());**

**}**

**break;**

**}**

**case "B" :**

**{**

**Scanner s4=new Scanner(System.in);**

**System.out.println("Enter the File Name to be deleted:");**

**String filename1=s4.next();**

**FileOperations.delete(filename1);**

**break;**

**}**

**case "C" :**

**{**

**Scanner s4= new Scanner (System.in);**

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

**String filename2=s4.next();**

**FileOperations.search(filename2);**

**List l1 = FileOperations.readFileInList(filename2);**

**Iterator<String> itr1 = l1.iterator();**

**while (itr1.hasNext())**

**{**

**System.out.println("The content of the file is:"+itr1.next());**

**}**

**break;**

**}**

**case "D" : {**

**System.out.println("Showing main menu");**

**showMainMenu();**

**break;**

**}**

**default :**

**System.out.println("Invalid Input. Please enter A, B, C or D");**

**}**

**showSubMenu();**

**}**

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

**{**

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

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

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

**System.out.println("Developed by: Atreyee Mondal");**

**Test obj= new Test();**

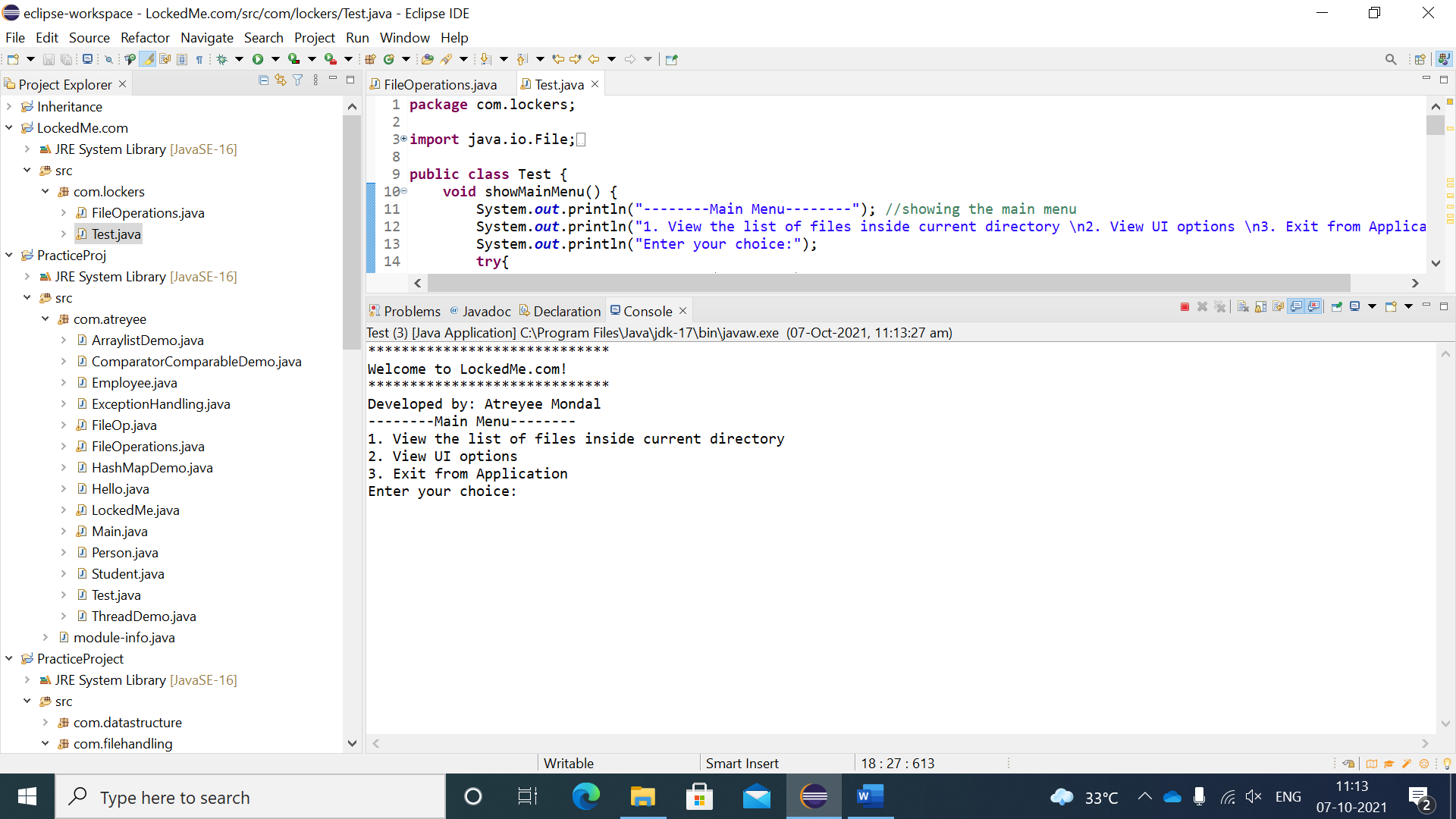
**obj.showMainMenu();**

**}**

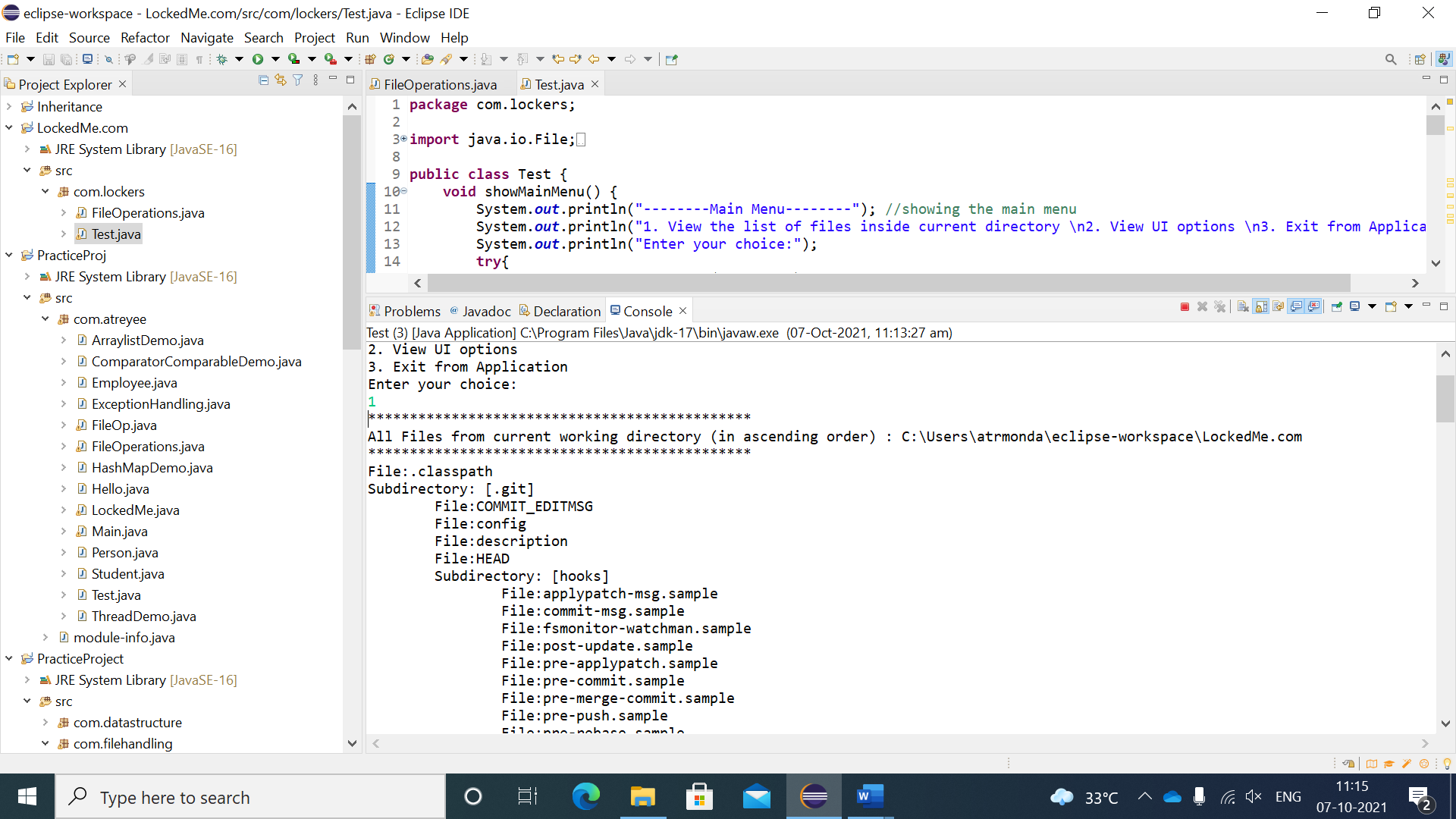
**}**

* **Stepwise Output Screenshot**

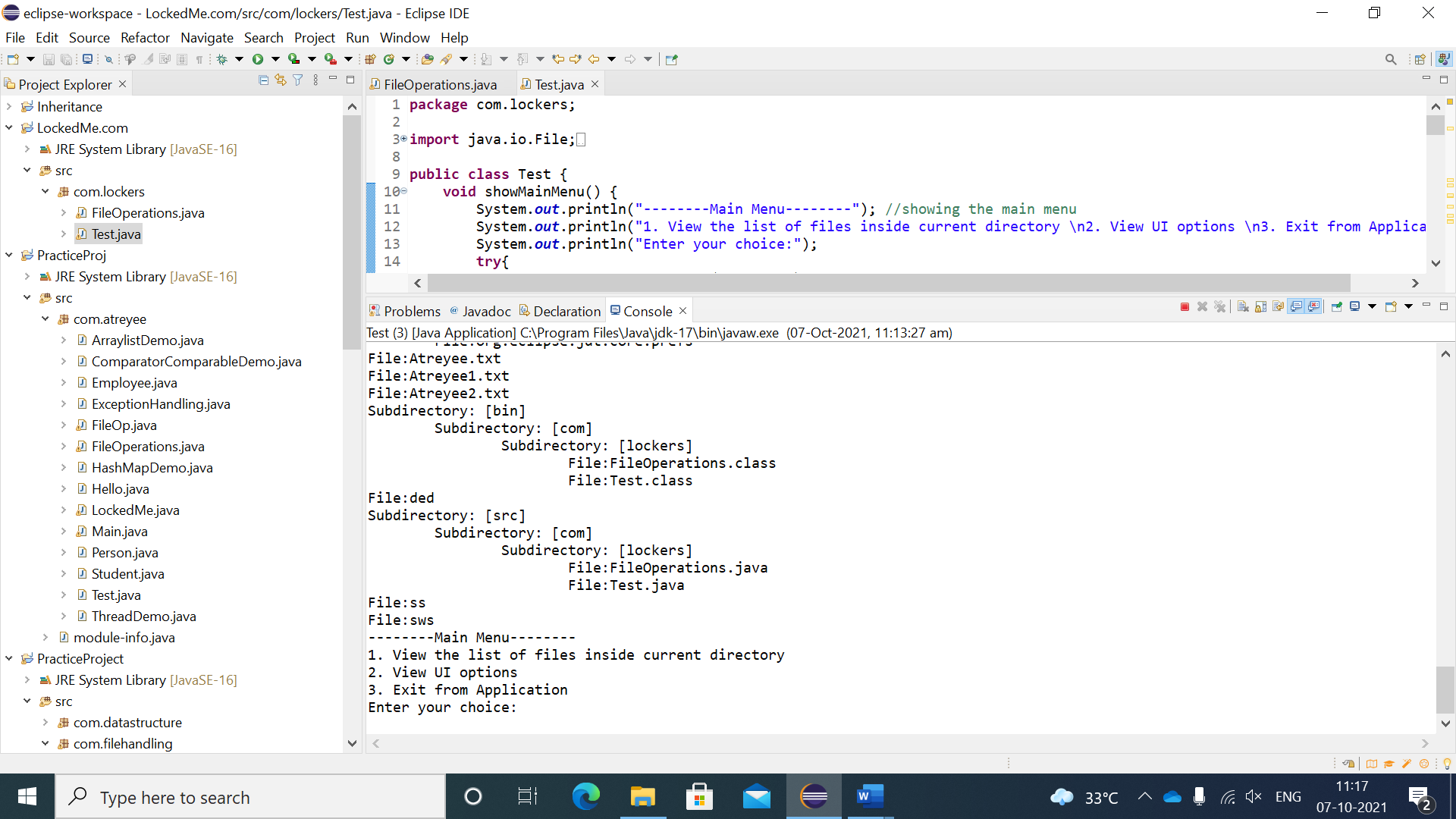
Step 1: Run the java application



Step 2.a: If user choose option1 from the main menu, it will show all the files, subdirectories and files inside each subdirectory in ascending order.

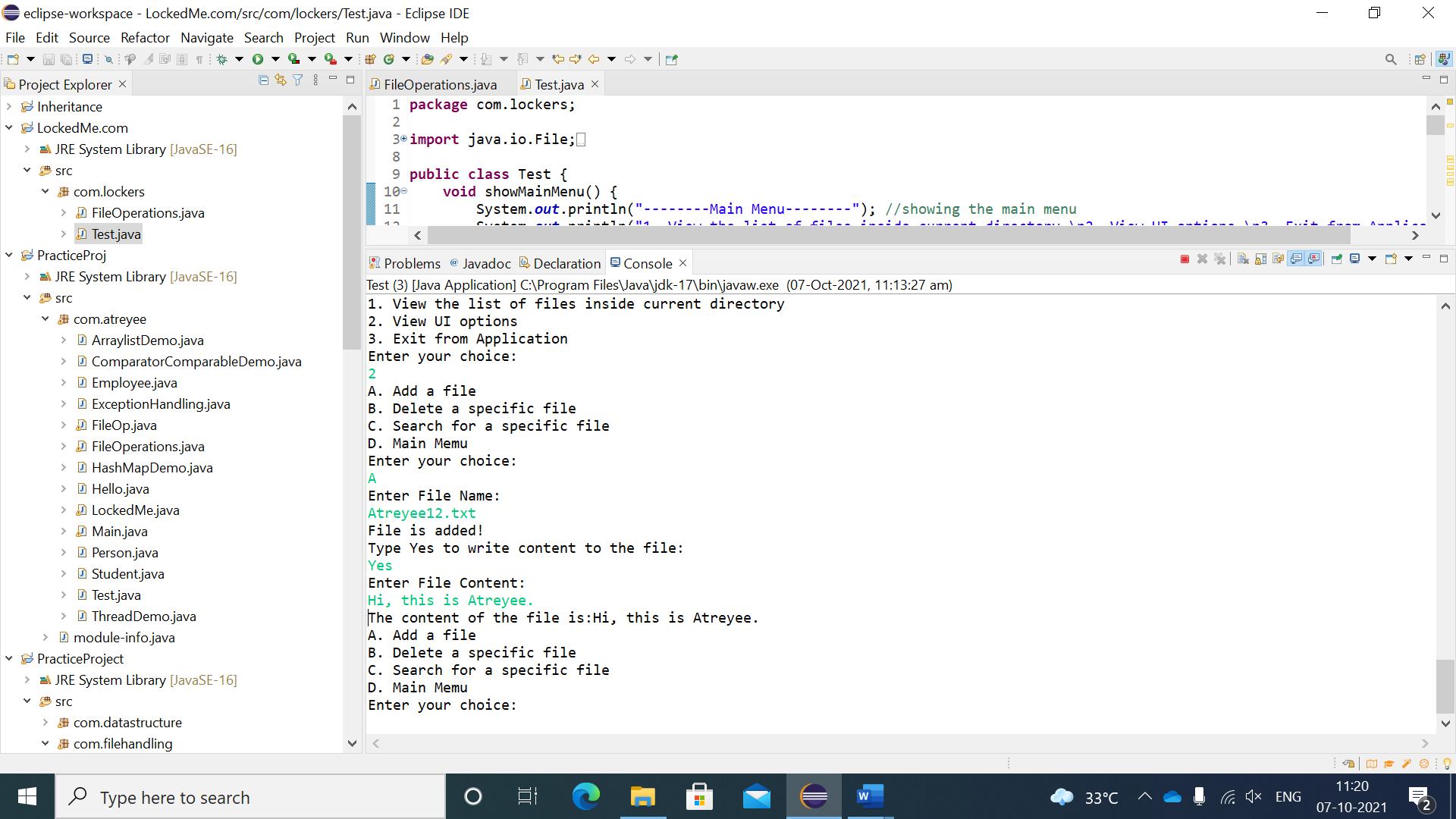


Step 2.b: From this option, control navigate back to the main menu.

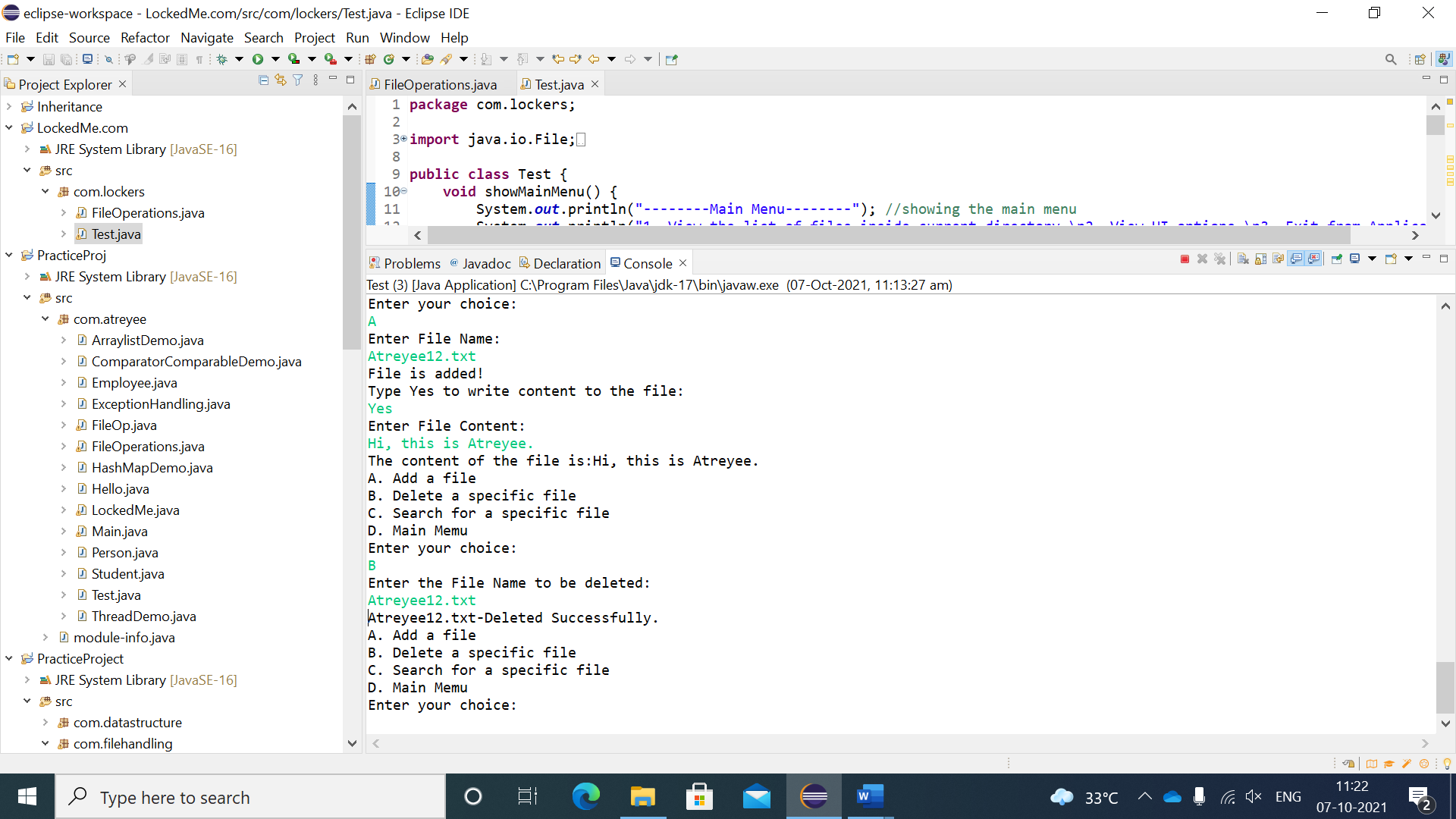


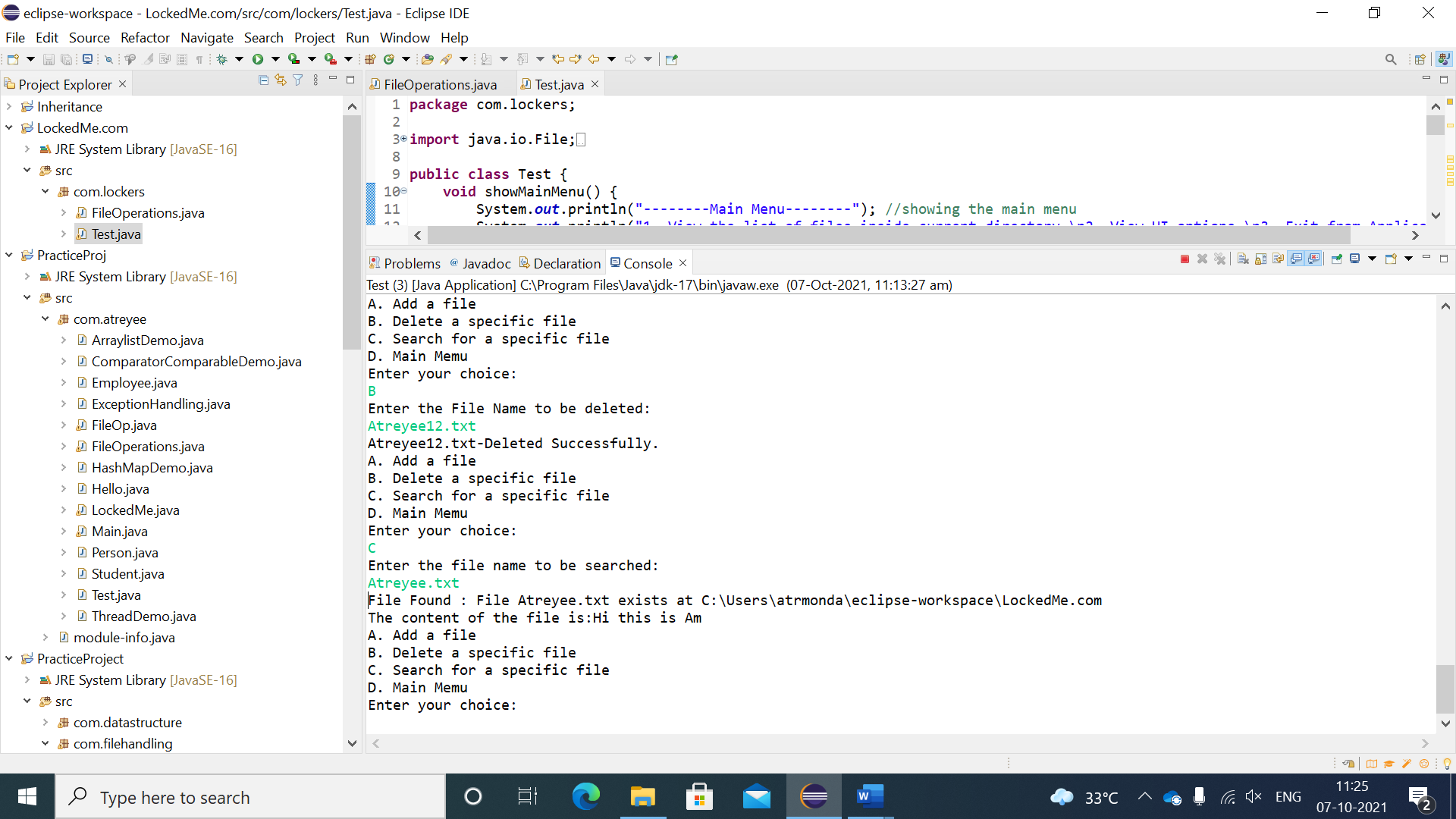
Step 3: If user choose option2 from the main menu, it will show another UI for adding, deleting and searching for a user specified file. It also shows an option to navigate back to the main menu.

1. Add a file to the current directory. If the file already exists, it will show “file already exist” else create the file and ask user to write content. Then show the content of the file. After creating the file, it will navigate back to the sub menu.

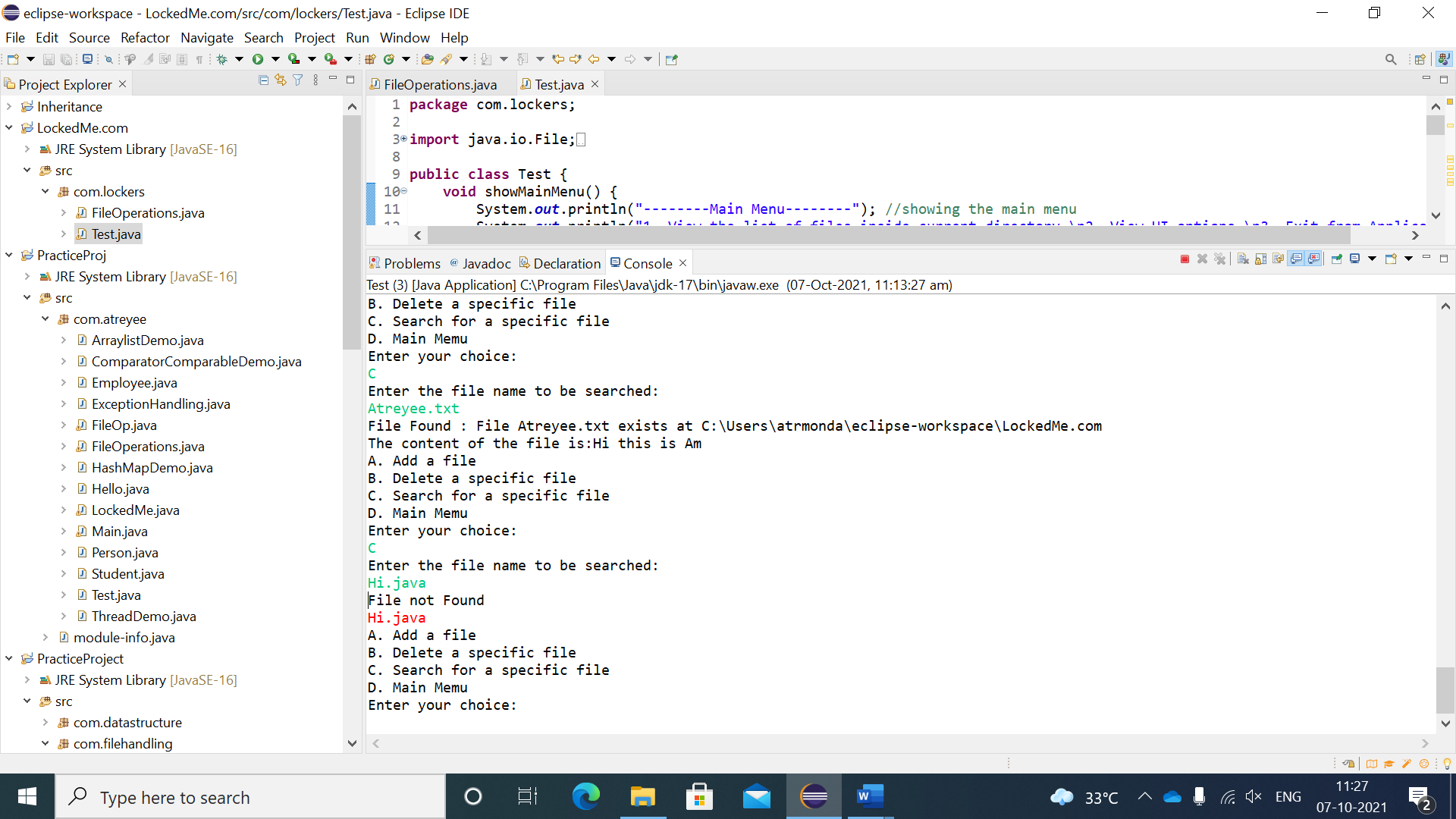


1. Delete a user specified file from the current directory, and if the file is not found, then it will show “File not Found”. After deleting the file, it will navigate back to the sub menu.

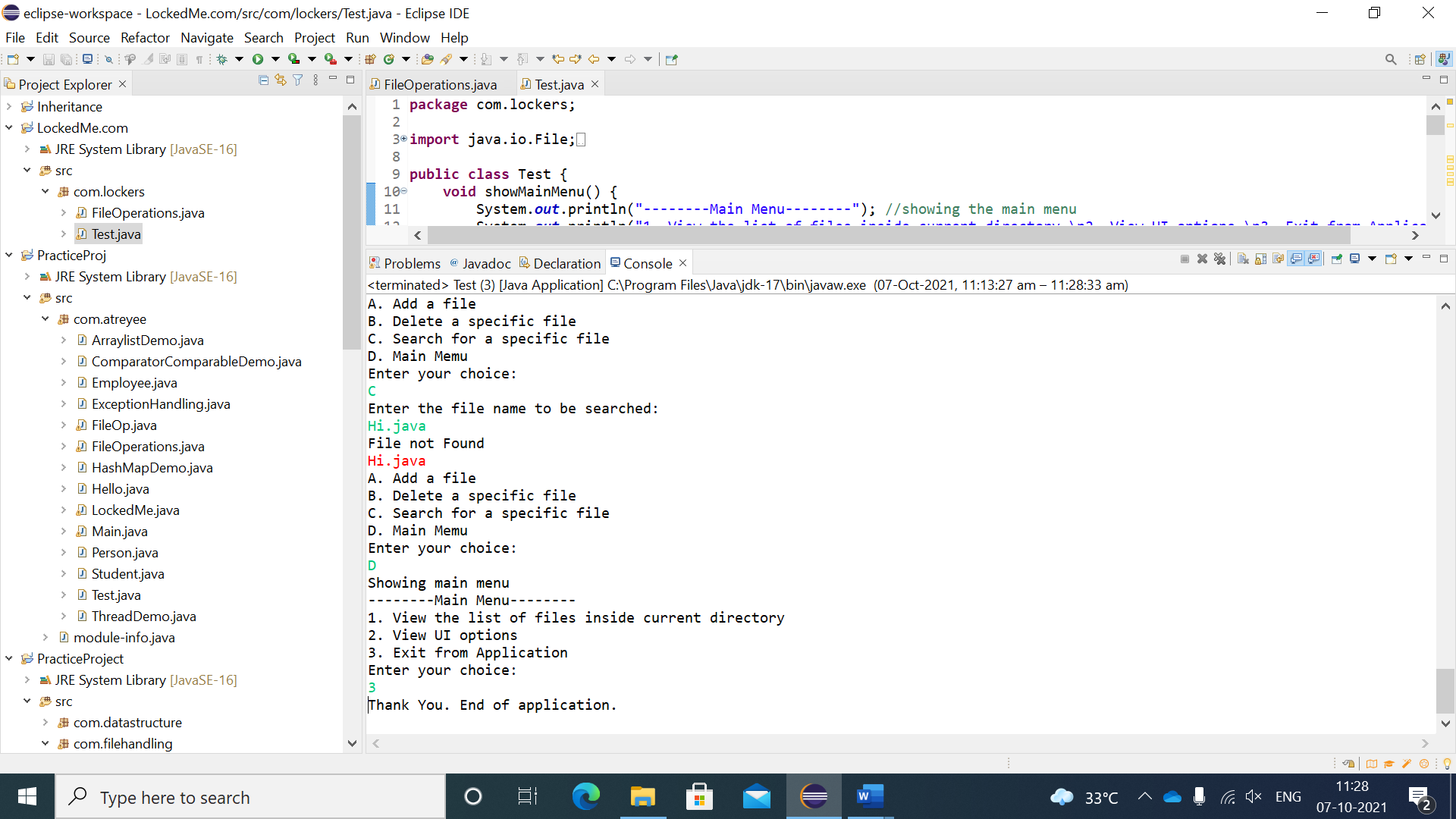


1. Search a user specified file and if file found then it will show the content of the file. If file is not found, it will show file not found. After searching the file, it will navigate back to the sub-menu.

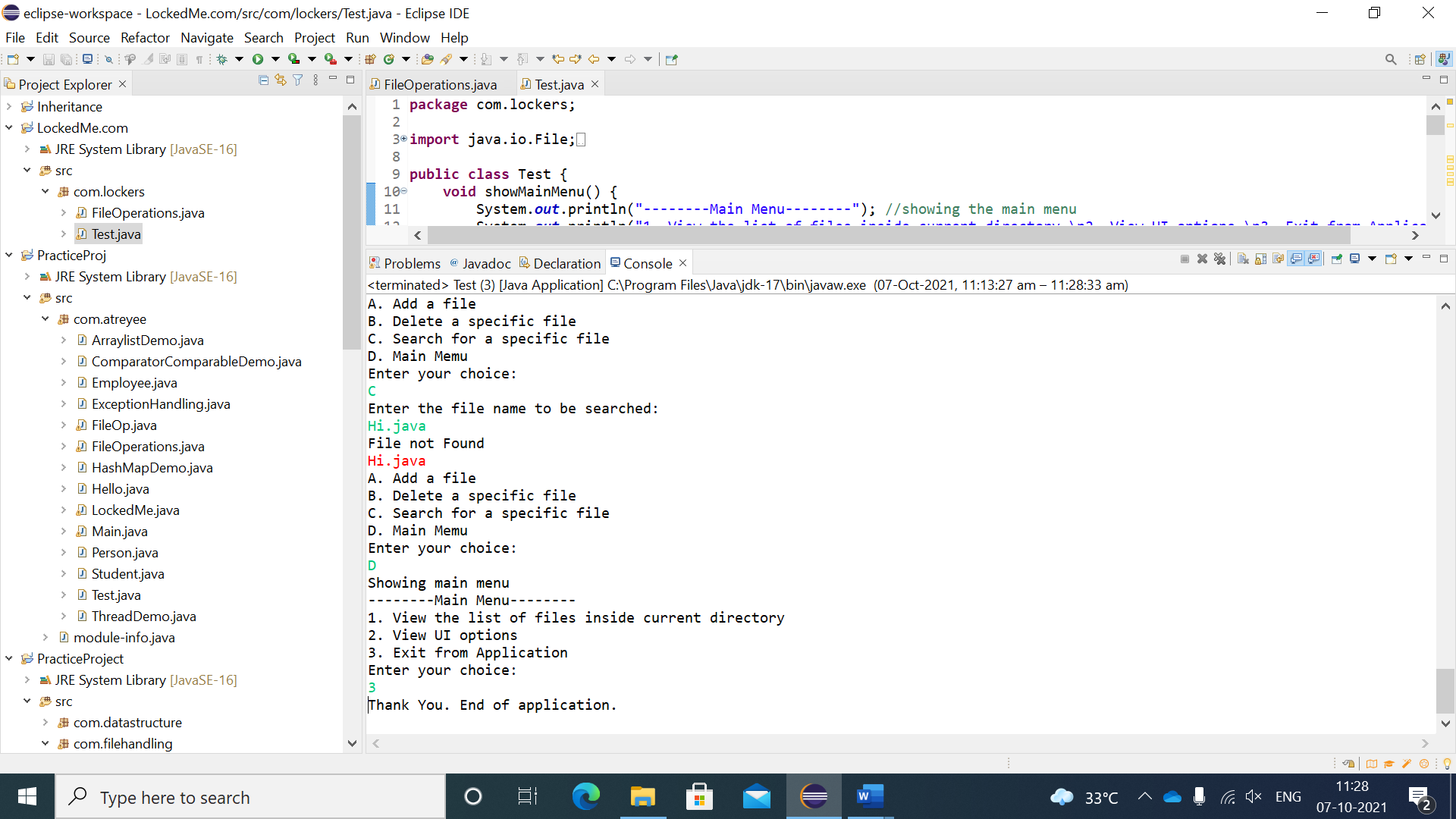
If file not found in search->



Step D: Navigate back to Main Menu:



Step 4: If the user enters oprion3 from the main menu, it will close the application.



* **Link to GitHub repository:**

<https://github.com/Atreyee28/LockedMe.com/tree/master/src/com/lockers>

**Procedure:**

* Open your command prompt and navigate to the folder where you have created your files.
  + **cd <folder name>**
* Initialize your repository using the following command:
  + **git init**
* Add all the files to your git repository using the following command:
  + **git add .**
* Commit the changes using the following command:
  + **git commit . -m “Changes have been committed.”**
* Push the files to the folder you initially created using the following command:
  + **git push -u origin master**
* **Conclusion:**

The application shown above is working properly without any error. Whenever the user enters any invalid input, it will not throw any exception or will not exit from the application. It will then prompt the user to enter correct value and show main menu. The UI is built to correctly take user input from command line. There are many core concepts such as exception handling, loops, collections, recursion, data structures etc. are used to implement the application. Finally, the application is pushed to GitHub repository to track the enhancement. All the business operations mentioned here are working correctly and outputs are attached.