**LockedMe – Virtual Key for Repositories**

This document contains sections as follow:

* **Sprint Planning and Task Completion**
* **Core concepts used in Project**
* **Flow chart of Application**
* **Product’s capabilities, appearance, and user interactions.**
* **Conclusions**

[GitHub Link:

Developers Details: Ashish Kumar Pathak

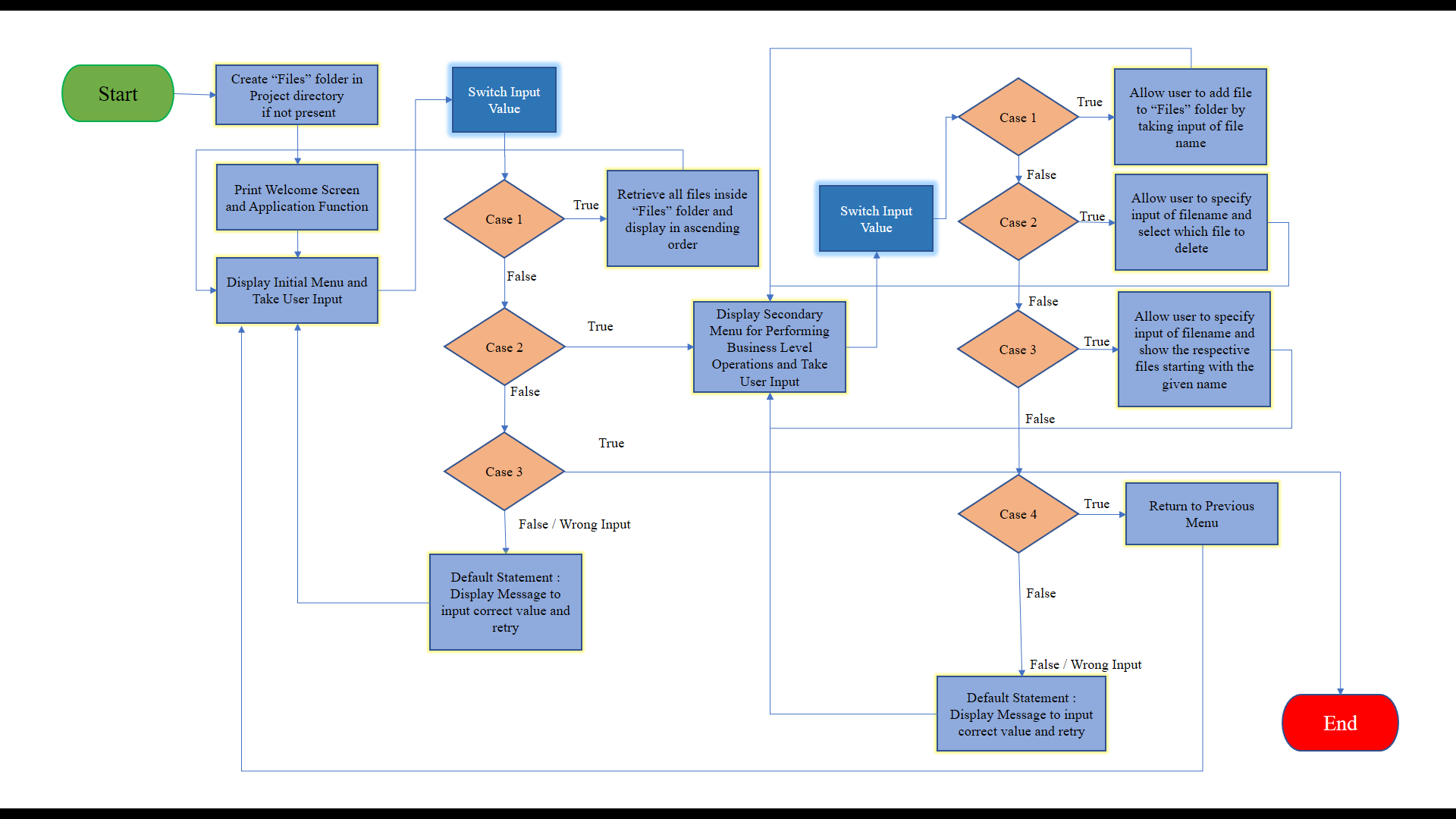
Designation: Java Developer

**Sprints Planning and Accomplishments**

The project is planned to completed in one sprint, The Task will be accomplished within sprint as below

1. Creation of flow of project
2. Initialization of git repository to track the changes as a development progresses.
3. Writing a Java based program to fulfil the requirements of the project.
4. Quality of Java based program is checked with different type of user inputs
5. Pushing the project to GitHub Repository.
6. Creating documentation for the project to enlighten the application capabilities, Appearance and user interactions.

**Flow Chart of Application**



## Explanation of the product capabilities, appearance, and user interactions

To Explain the product capabilities, there are sub- sections configured to highlight appearance and user interactions for the project.

1. Creation of a project in Eclipse IDE.
2. Writing a Java Program for entry point of the application i.e., (LockedMeMain.java)
3. Writing a Java Program, to display the Main Menu options for the user’s interactions i.e., (DisplayOptions.java)
4. Writing a Java Program, to handle Main Menu options available for the user i.e., (OperationHandling.java)
5. Writing a Java Program, to perform given input task as specified by the users i.e., (LockedMeFileHandling.java)
6. The project is pushed to GitHub Repository.

**Step 1:** Creation of project in Eclipse IDE.

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in any project name and click on “Finish.”
* Creating a package into scr folder with name as com.lockedme.application.
* Select your created package and go to File -> New -> Class.
* Enter **LockedMeMain** in any class name, check the checkbox “public static void main (String [] args)”, and click on “Finish.”

**Step 2: Writing a Java Program, for entry point of the application. (LockedMeMain.java)**

Coding:

**package** com.lockedme.application;

**public** **class** LockedMeMain {

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

//Files Folder is created if not exist in the project.

LockedMeFileHandling.*createFolderIfNotPresent*("Files");

DisplayOptions.*displayWelcomeScreen*();

OperationHandling.*mainMenu*();

}

}

**Step 3: Writing a Java Program, To Display the main menu options available for user interactions** i.e.**, (DisplayOptions.java).**

* Select your package and go to File -> New -> Class.
* Enter DisplayOptions in the class name and click on Finish.
* DisplayOptions contains methods as:
  1. Displaying Welcome Screen and developer details
  2. Displaying Main Menu
  3. Displaying Business Level Operation menu.

**Step 3.1:** Writing a method to display Welcome Screen and Developer Details.

Coding:

**public** **static** **void** displayWelcomeScreen() {

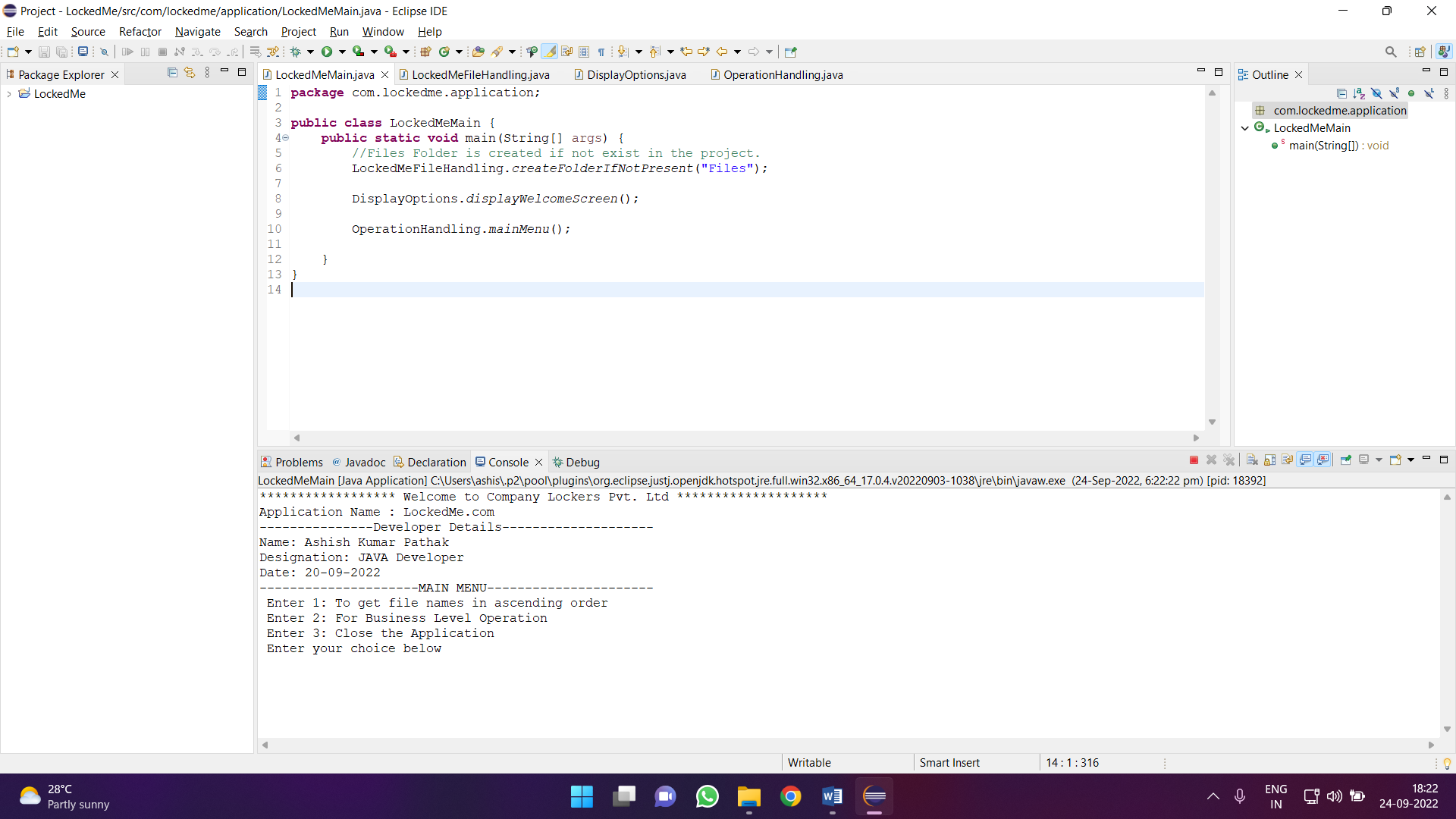
System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Welcome to Company Lockers Pvt. Ltd \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

+ "Application Name : LockedMe.com\n" + "---------------Developer Details--------------------\n"

+ "Name: Ashish Kumar Pathak \n" + "Designation: JAVA Developer\n" + "Date: 20-09-2022");

}

**Output:**



**Step 3.2:** Writing a method to display main menu**.**

Coding:

**public** **static** **void** displayMainMenu() {

System.***out***.println(

"---------------------MAIN MENU----------------------\n"

+"\*\*\*\*Please select any number from below menu and press

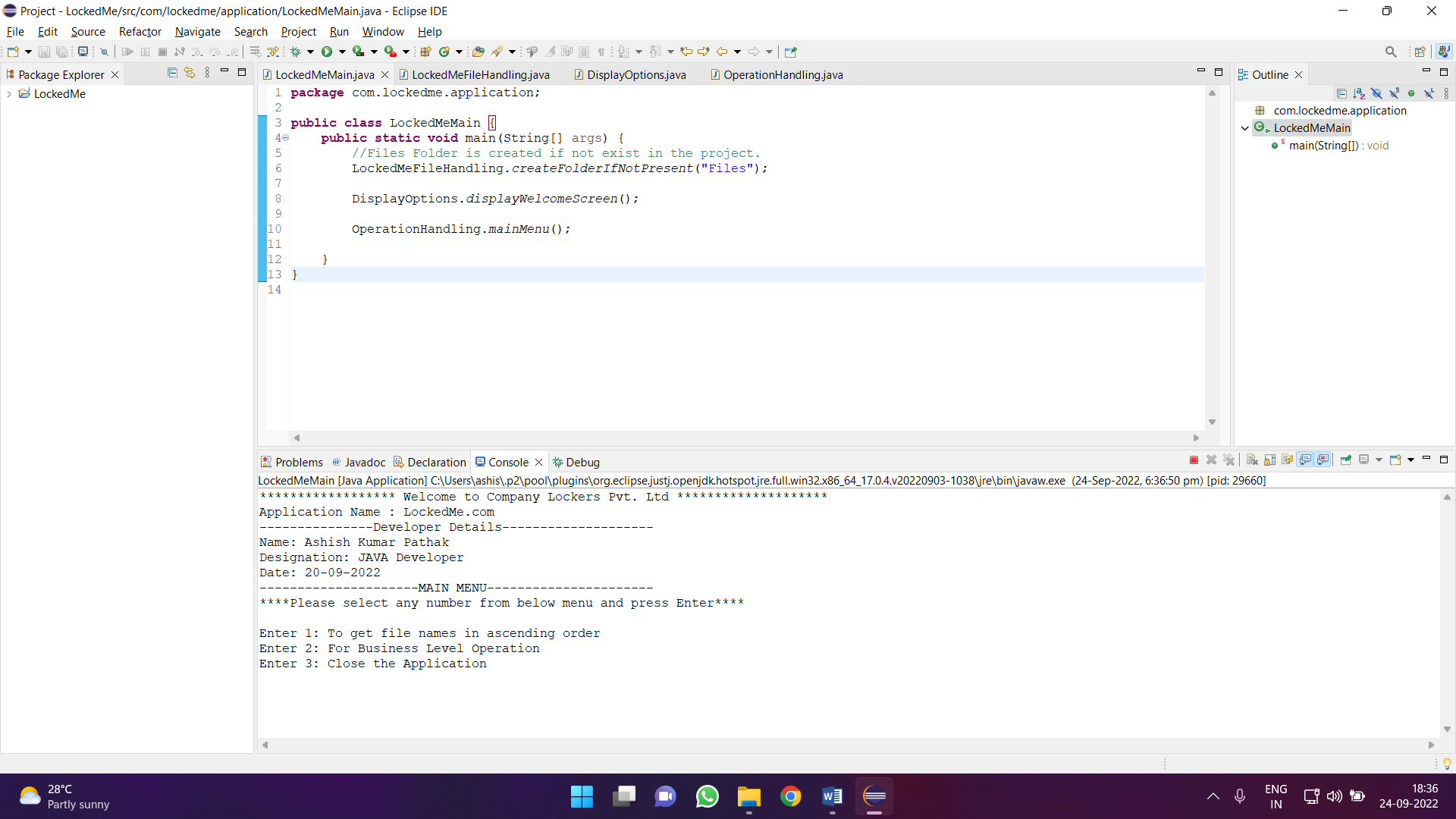
Enter\*\*\*\*\n\n"+ "Enter 1: To get file names in ascending

order\n"+ "Enter 2: For Business Level Operation\n"

+ "Enter 3: Close the Application\n");

}

**Output:**



**Step 3.3:** Writing a method display business level operations menu.

Coding:

**public** **static** **void** displayBusinessLevelOperationMenu() {

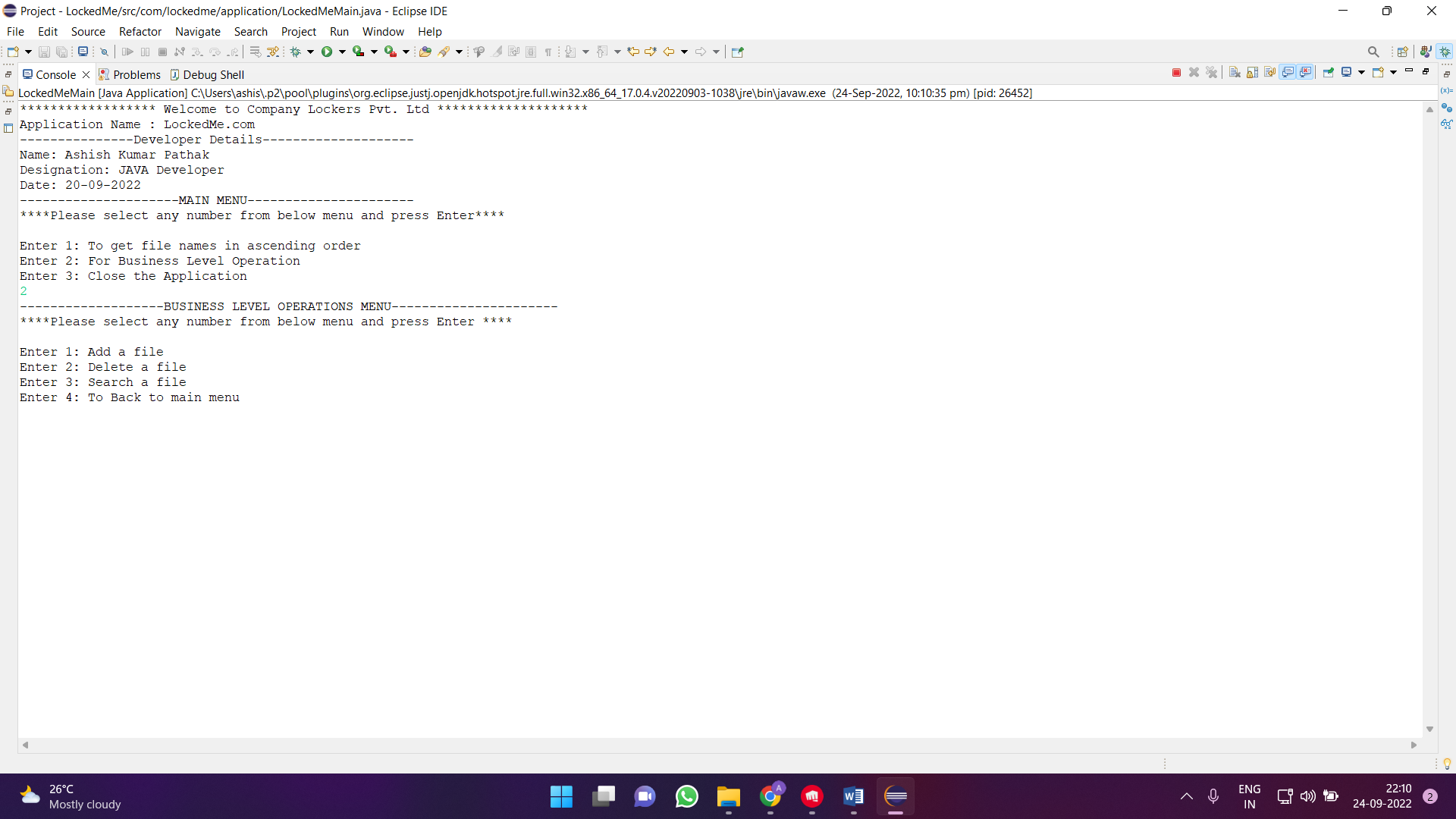
System.***out***.println("-------------------BUSINESS LEVEL OPERATIONS MENU----------------------"

+ "\n\*\*\*\*Please select any number from below menu and press Enter \*\*\*\*\n\nEnter 1: Add a file"

+ "\nEnter 2: Delete a file\nEnter 3: Search a file\nEnter 4: To Back to main menu");

}

**Output:**



**Step 4: Writing a program to handle Main Menu Options for User’s Interactions** (OperationHandling.java)

* Select your package and right click go to File-> New -> Class.
* Enter OperationHandling in class name and click on “Finish”.
* OperationHandling Class Consists methods for:

**4.1** Handling input given by user in Main Menu

**4.2** Handling input given by user in Business Level Operation Menu.

**Step 4.1:** Writing a method to handle the user given input as Initial Menu

Coding:

**public** **static** **void** mainMenu() {

// Do - While loop as True for Application to run until user give input to exit

// the Application

**boolean** isApplicationRunning = **true**;

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

**do** {

**try** {

DisplayOptions.*displayMainMenu*();

**int** choice = scan.nextInt();

**switch** (choice) {

**case** 1:

LockedMeFileHandling.*displayFilesInAscendingOrder*("Files");

**break**;

**case** 2:

OperationHandling.*businessLevelOperation*();

**break**;

**case** 3:

System.***out***.println("Program exited successfully.");

isApplicationRunning = **false**;

scan.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please select a valid option from above.");

}

} **catch** (Exception e) {

System.***out***.println(e.getClass().getName());

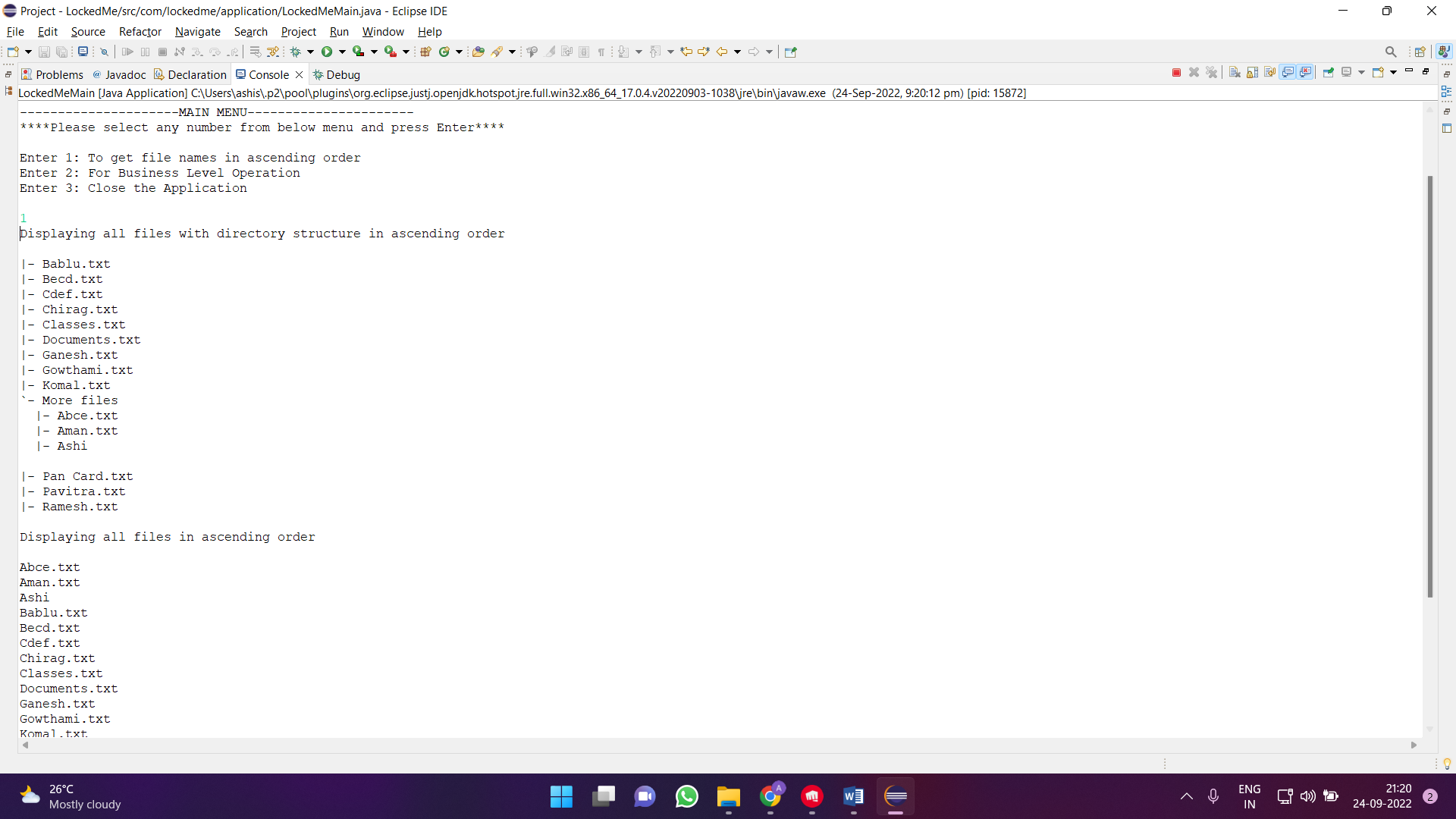
*mainMenu*();

}

} **while** (isApplicationRunning == **true**);

}

**OUTPUT:**



**Step 4.2:** Writing a method to handle user input for business level operations menu.

Coding:

**public** **static** **void** businessLevelOperation() **throws** IOException {

**boolean** isApplicationRunning = **true**;

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

**do** {

**try** {

DisplayOptions.*displayBusinessLevelOperationMenu*();

// Switch is used to perform different type of task - input such given by the

// user.

**int** businessLevelOperationChoice = sc.nextInt();

**switch** (businessLevelOperationChoice) {

**case** 1: {

LockedMeFileHandling.*createFile*();

**break**;

}

**case** 2: {

LockedMeFileHandling.*deleteFile*();

**break**;

}

**case** 3: {

LockedMeFileHandling.*displayFile*();

**break**;

}

**case** 4: {

System.***out***.println("Back to Main Menu is Successful");

isApplicationRunning = **false**;

**return**;

}

**default**:

System.***out***.println("Please enter valid option from above.");

}

} **catch** (Exception e) {

System.***out***.println(e.getClass().getName());

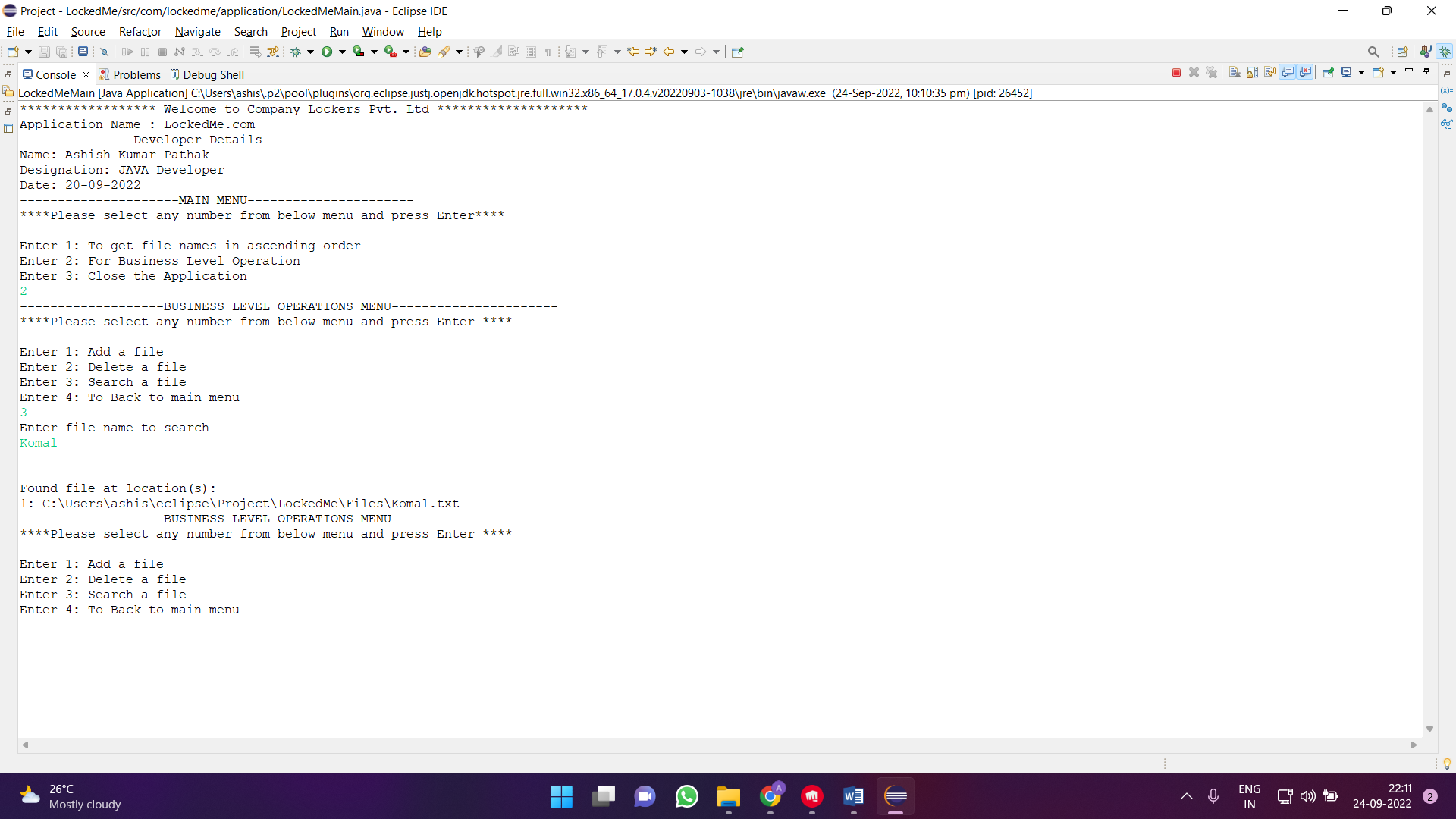
*businessLevelOperation*();

}

} **while** (isApplicationRunning == **true**);

}

**Output:**



**STEP 5:** Writing a Java program, to perform the given input task by the user LockedMeFileHandling.java.

* Select your package and go to File -> New -> Class.
* Enter LockedMeFileHandling in class name and click on ‘Finish’.
* LockedMeFileHandling Class consist of methods as:

5.1. Creation of ‘Files’ Folder in the project if it’s not exist.

5.2. Displaying the all files from ‘Files’ folder in ascending order and there directory.

5.3. Creating a file as specified by user.

5.4. Deletion of a file as specified by the user.

5.5. Searching of the file from the ‘Files’ folder.

**Step 5.1.:** Writing a method to create ‘Files’ Folder in project if not exists.

Coding:

**public** **static** **void** createFolderIfNotPresent(String folderName) {

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

// If file doesn't exist, create the "File" folder

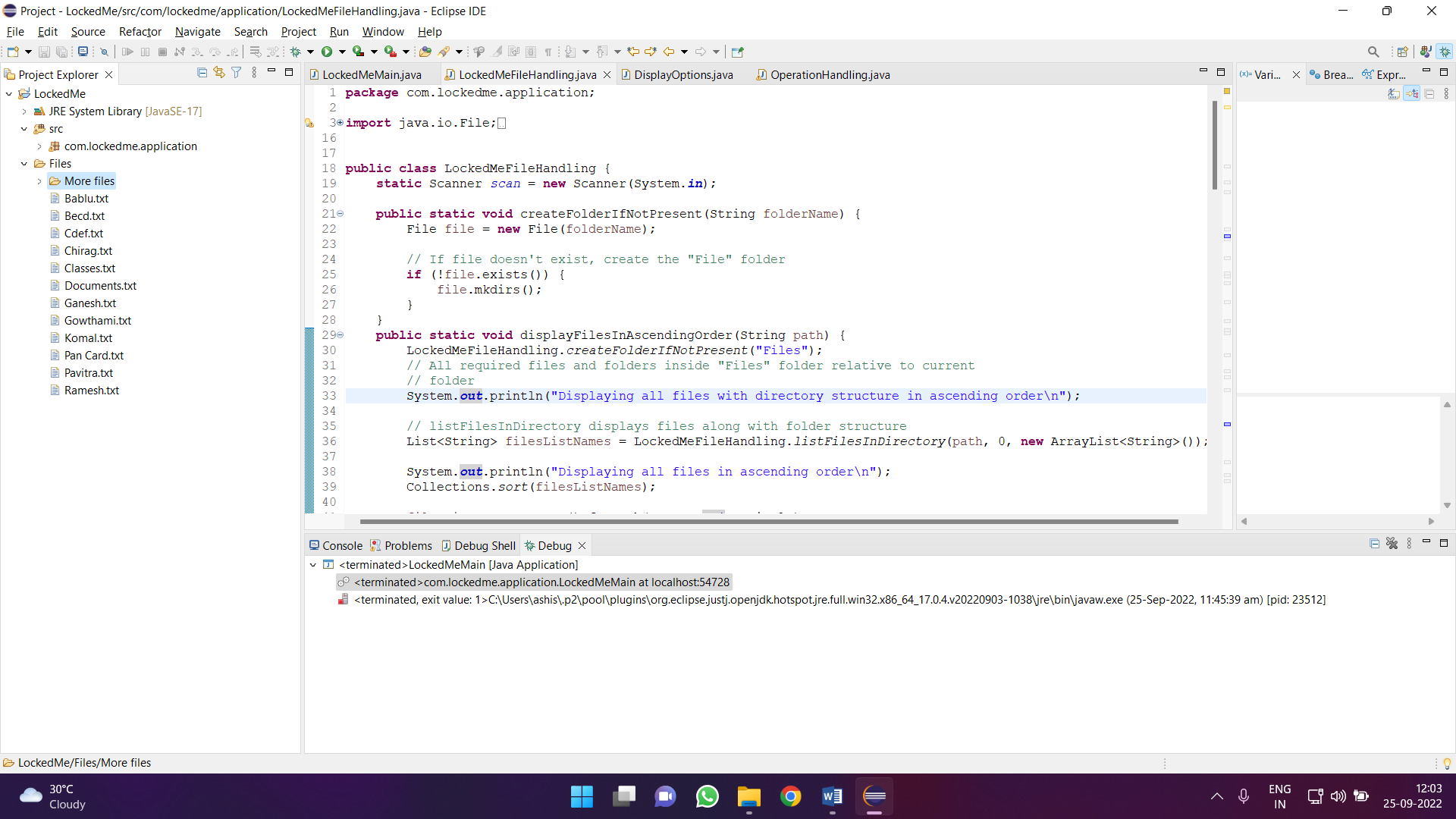
**if** (!file.exists()) {

file.mkdirs();

}

}

**Output:**



**Step 5.2.:** Writing a method to get the all files from ‘Files’ Folder in ascending order

**Coding:**

**public** **static** **void** displayFilesInAscendingOrder(String path) {

LockedMeFileHandling.*createFolderIfNotPresent*("Files");

// All required files and folders inside "Files" folder relative to current

// folder

System.***out***.println("Displaying all files with directory structure in ascending order\n");

// listFilesInDirectory displays files along with folder structure

List<String> filesListNames = LockedMeFileHandling.*listFilesInDirectory*(path, 0, **new** ArrayList<String>());

System.***out***.println("Displaying all files in ascending order\n");

Collections.*sort*(filesListNames);

filesListNames.stream().forEach(System.***out***::println);

}

**public** **static** List<String> listFilesInDirectory(String path, **int** blankSpaceCount, List<String> fileListNames) {

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

File[] files = dir.listFiles();

List<File> filesList = Arrays.*asList*(files);

Collections.*sort*(filesList);

**if** (files != **null** && files.length > 0) {

**for** (File file : filesList) {

System.***out***.print(" ".repeat(blankSpaceCount \* 2));

**if** (file.isDirectory()) {

System.***out***.println("``-- " + file.getName());

// Recursively indent and display the files

fileListNames.add(file.getName());

*listFilesInDirectory*(file.getAbsolutePath(), blankSpaceCount + 1, fileListNames);

} **else** {

System.***out***.println("|- " + file.getName());

fileListNames.add(file.getName());

}

}

} **else** {

System.***out***.print(" ".repeat(blankSpaceCount \* 2));

System.***out***.println("|- Empty Directory");

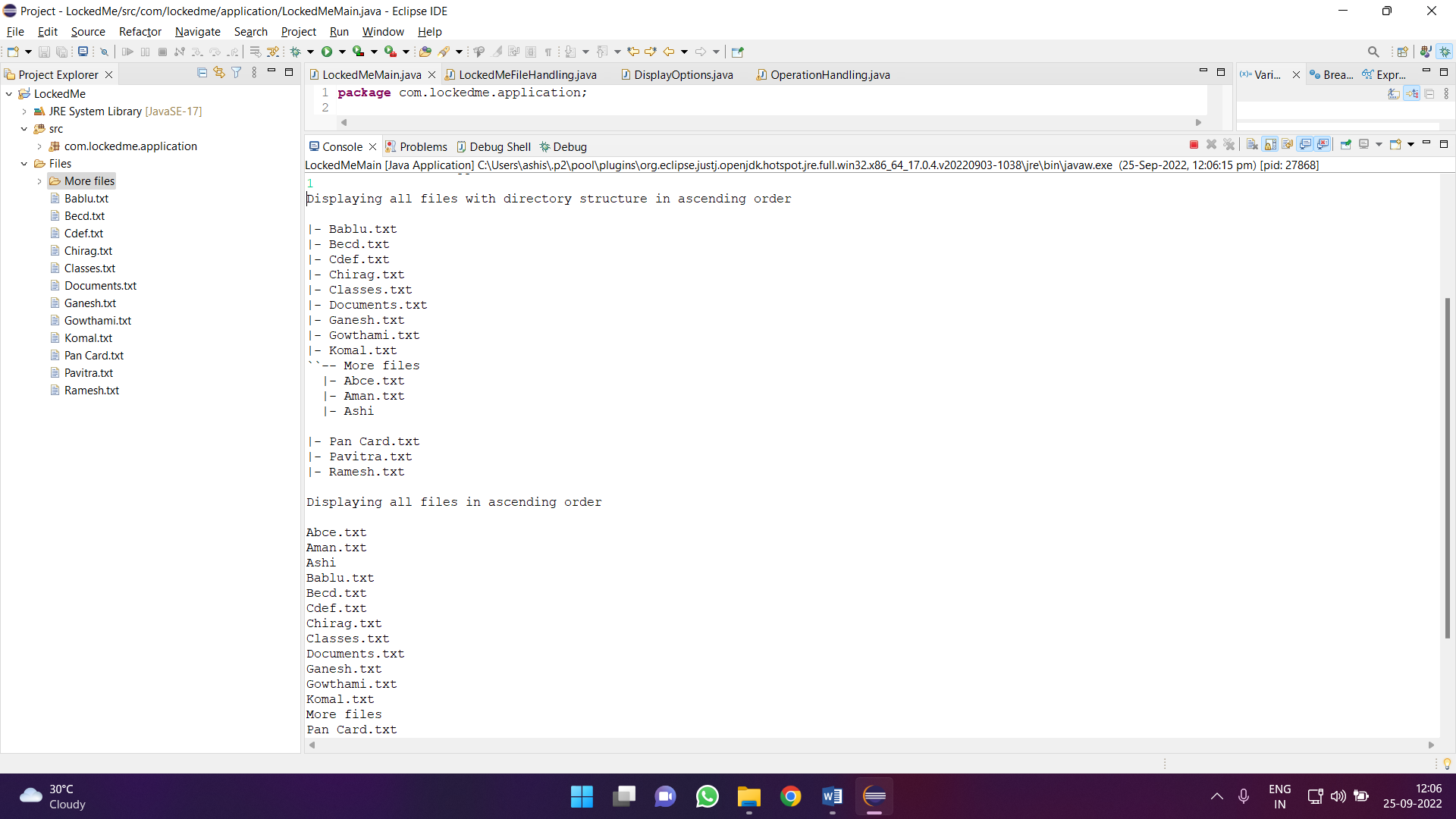
}

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

**return** fileListNames;

}

**Output:**



**Step 5.3.:** Writing a method to create a File as specified by the user.

**Coding:**

**public** **static** **void** createFile() {

LockedMeFileHandling.*createFolderIfNotPresent*("Files");

System.***out***.println("Enter your file name");

String fileName = *scan*.next();

Path pathToFile = Paths.*get*("./Files/" + fileName);

**try** {

Files.*createDirectories*(pathToFile.getParent());

Files.*createFile*(pathToFile);

System.***out***.println(fileName + " created successfully\n");

} **catch** (IOException e) {

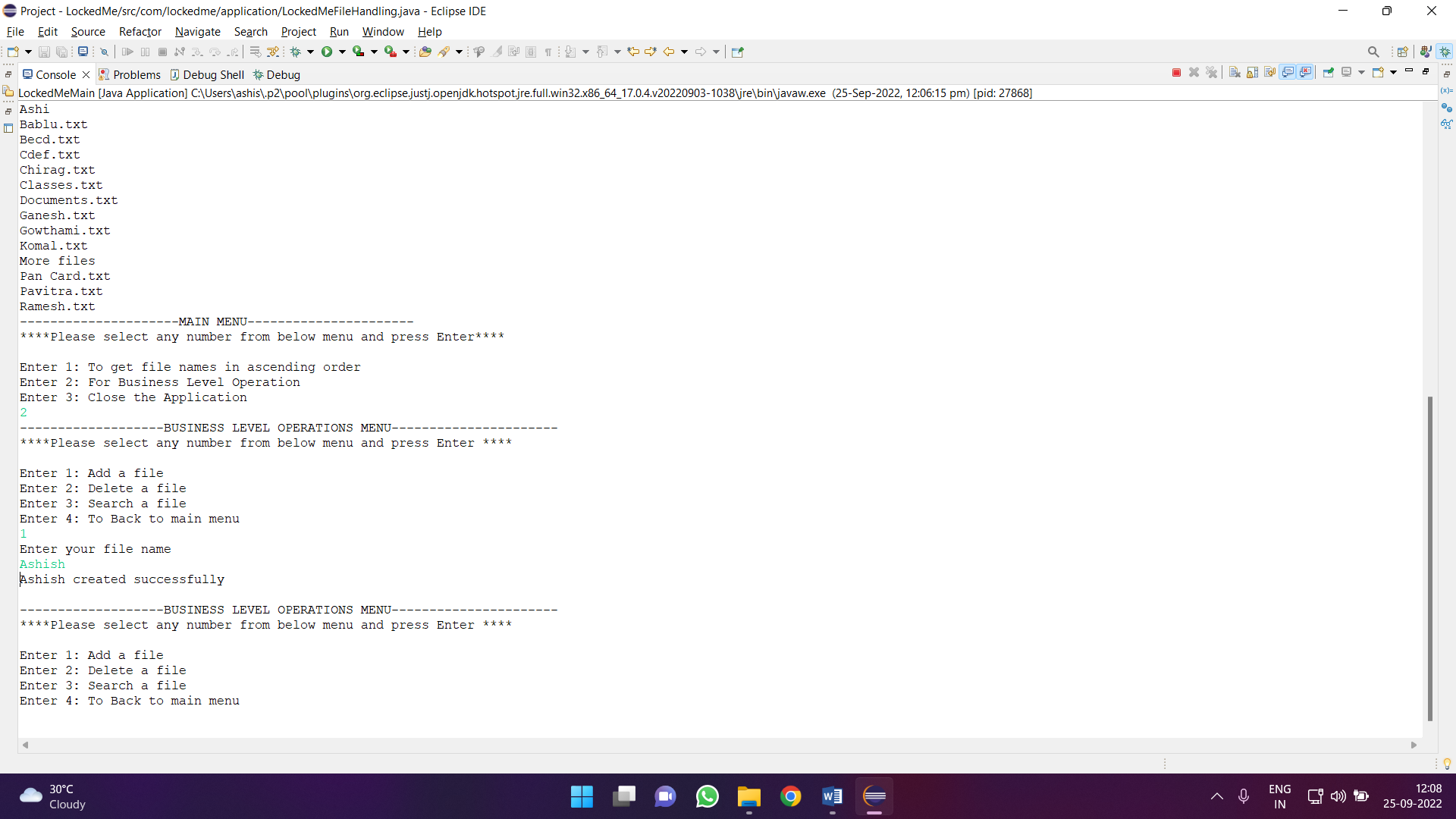
System.***out***.println("Failed to create file " + fileName);

System.***out***.println(e.getClass().getName());

}

}

**Output:**



**Step 5.4.:** Writing a method to delete a File as specified by the user.

**Coding:**

**public** **static** **void** deleteFile() {

System.***out***.println("Enter file name for deletion");

String deleteFileName = *scan*.next();

File deleteFileDirectory = **new** File(System.*getProperty*("user.dir")+"//Files//" + deleteFileName);

**if** (deleteFileDirectory.delete())

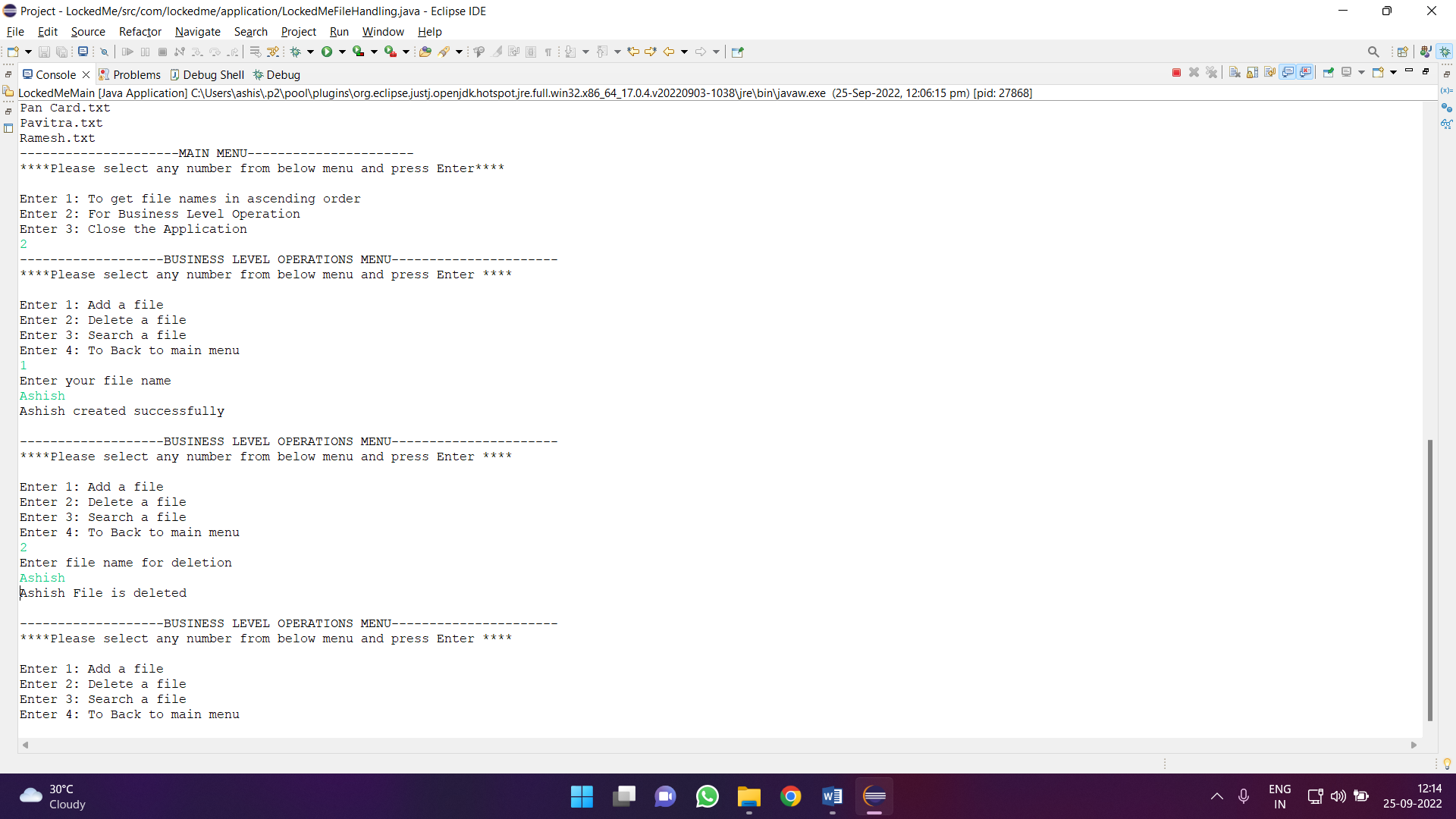
System.***out***.println(deleteFileName + " File is deleted\n");

**else**

System.***out***.println(deleteFileName + " is not deleted as file is not found in directory\n");

}

**Output:**



**Step 5.5.:** Writing a method to search a File as specified by the user.

**Coding:**

**public** **static** **void** displayFile() {

List<String> fileListNames = **new** ArrayList<>();

File searchFileDirectory = **new** File(System.*getProperty*("user.dir")+"//Files//");

String path = searchFileDirectory.toString();

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

String searchFileName = *scan*.next();

LockedMeFileHandling.*searchFile*(path, searchFileName, fileListNames);

**if** (fileListNames.isEmpty()) {

System.***out***.println("\n\nCouldn't find any file with given file name \"" + searchFileName + "\"\n\n");

} **else** {

System.***out***.println("\n\nFound file at location(s):");

List<String> files = IntStream.*range*(0, fileListNames.size())

.mapToObj(index -> (index + 1) + ": " + fileListNames.get(index)).collect(Collectors.*toList*());

files.forEach(System.***out***::println);

}

}

**public** **static** **void** searchFile(String path, String fileName, List<String> fileListNames) {

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

File[] files = dir.listFiles();

List<File> filesList = Arrays.*asList*(files);

**if** (files != **null** && files.length > 0) {

**for** (File file : filesList) {

**if** (file.getName().startsWith(fileName)) {

fileListNames.add(file.getAbsolutePath());

}

//To check if file is available in inside other directory.

**if** (file.isDirectory()) {

*searchFile*(file.getAbsolutePath(), fileName, fileListNames);

}

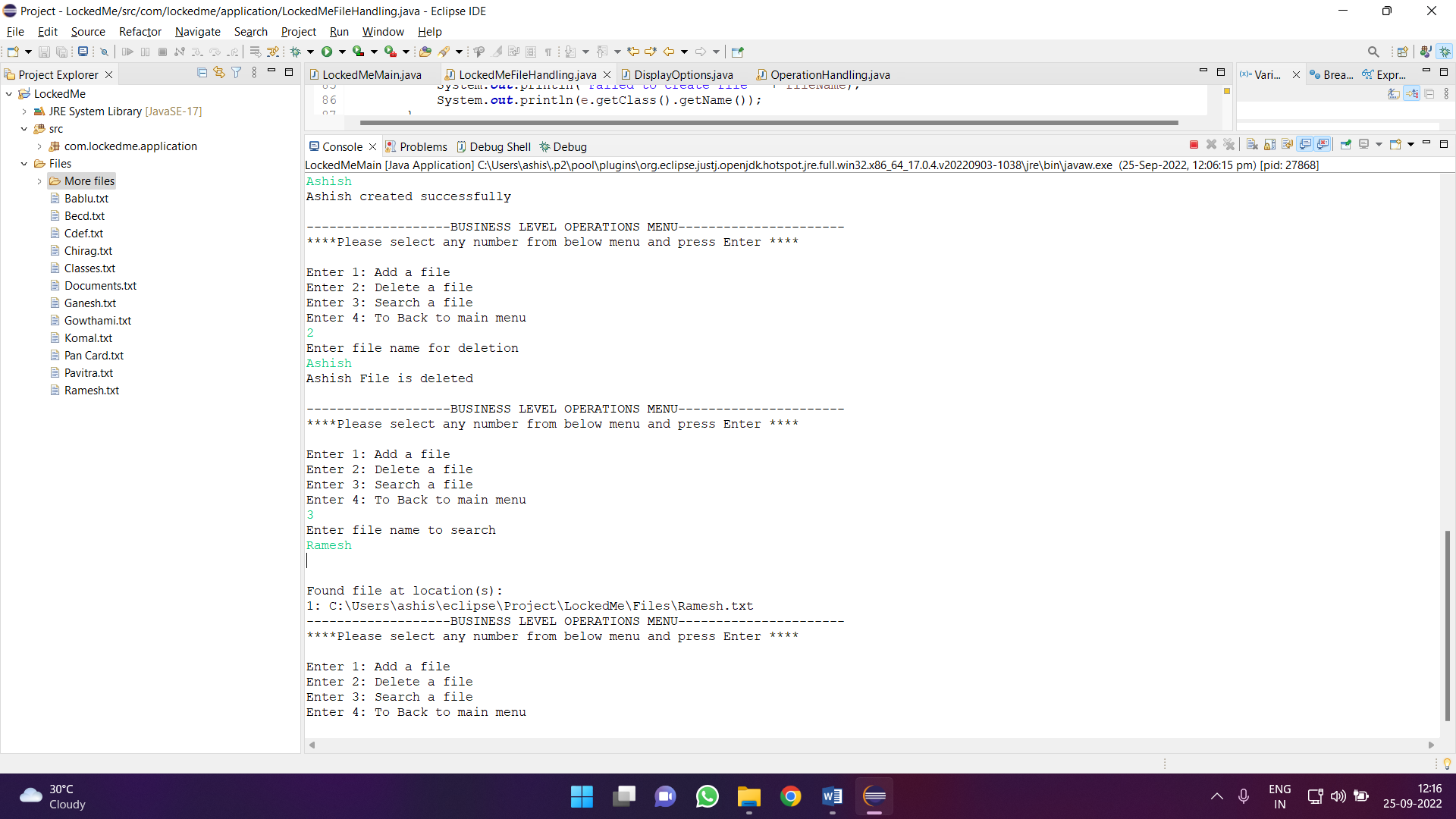
}

}

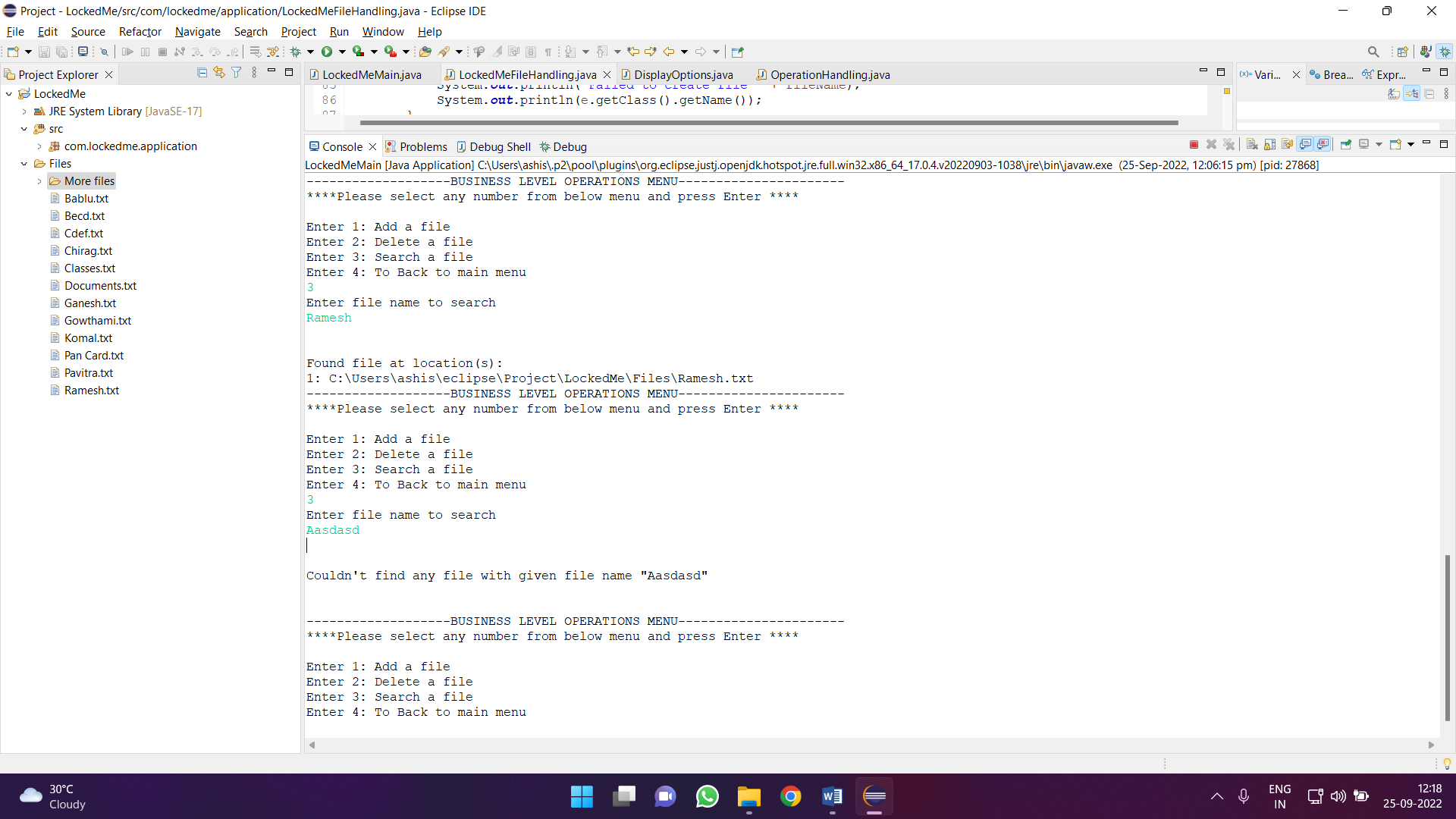
}

**Output:**

**If file is found**



**If file is not found**



**STEP 5:** The project is pushed to GitHub repository.

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize 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 <commit message>**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**

## Unique Selling Points of the Application

1. Application is designed in such a way that it keeps on running and takes user inputs even after there is a exception occurs. Termination of the application is done by user given option is selected.
2. The Application is designed such a way that it can take input from user for performing different kinds of task such as displaying of the files and then Business Level Operations such as add, delete, searching of files.
3. User can give input for search option if the file is not present user gets the message to enter with case sensitivity
4. Deletions of the file is done as the input given to the program. Particular file which has been given is deleted. If not deleted the message pop-up as file cannot be found to delete.
5. 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.
6. When the option to retrieve files in ascending order is selected, user is displayed with all file names and files within folder also displays
7. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code. Application has been developed keeping in mind that there should be very less “hardcoding” of data.

**Conclusions:**

* Further enchantment can be made as files can be stored in database for better retrieve and use of the user interactions.
* The Data stored into the database can be secured and While retrieval we can use database query to select the particular information.