ASSESSMENT

**Phase 1: Implement OOPS using JAVA with Data Structures and beyond**

Submitted By: MEGHAJ KUMAR MALLICK

Github Repository Link: <https://github.com/MEGHAJ17/Project.git>

E-Mail Id:meghajkumarmallick@gmail.com

Contact No: 7004896040

TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| SR NO | CONTENT | PAGE NO. |
| 1 | ACKNOWLEDMENT | 3 |
| 2 | PROBLEM STATEMENT | 4-6 |
| 3 | USER STORY | 7 |
| 4 | SPRINTS & PLANNING | 7-8 |
| 5 | AIM | 8 |
| 6 | OBJECTIVE | 8 |
| 7 | SYSTEM REQUIREMENTS | 8-9 |
| 8 | DATA FLOW DIAGRAM (LEVEL 0 DIAGRAM) | 9 |
| 9 | PROPOSED METHOD | 10 |
| 10 | JAVA CODE WITH OUTPUT | 10-16 |
| 11 | REFERENCES | 19 |

1. ACKNOWLEDMENT

I would like to thank Simplilearn who gave me this opportunity to work on this project. I got to learn a lot from this project about implantation of object oriented programming structure using data structure.

At last, I would like to extend my heartfelt thanks to my parents because without their help this project would not have been successful. Finally, I would like to thank my dear friends who have been with me all the time.

I would like to express my special thanks to our mentor Deepak Sir for his time and efforts he provided throughout the online session. Your useful advice and suggestions were really helpful to me during the project’s completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Signature

Meghaj Kumar Mallick

1. PROBLEM STATEMENT

**Virtual Key for Your Repositories**

Project 1

DESCRIPTION

**Project objective:**

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**Background of the problem statement:**

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You’re asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you’re asked to present the following in the next 15 working days (3 weeks):

* Specification document - Product’s capabilities, appearance, and user interactions
* Number and duration of sprints required
* Setting up Git and GitHub account to store and track your enhancements of the prototype
* Java concepts being used in the project
* Data Structures where sorting and searching techniques are used.
* Generic features and three operations:
  + 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
  + The goal of the company is to deliver a high-end quality product as early as possible.

**The flow and features of the application:**

* Plan more than two sprints to complete the application
* Document the flow of the application and prepare a flow chart
* List the core concepts and algorithms being used to complete this application
* Code to display the welcome screen. It should display:
  + Application name and the developer details
  + The details of the user interface such as options displaying the user interaction information
  + Features to accept the user input to select one of the options listed
  + The first option should return the current file names in ascending order.
  + The root directory can be either empty or contain few files or folders in it
  + The second option should return the details of the user interface such as options displaying the following:
  + Add a file to the existing directory list
    - You can ignore the case sensitivity of the file names
  + Delete a user specified file from the existing directory list
    - You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
    - Return a message if FNF (File not found)
  + Search a user specified file from the main directory
    - You can add the case sensitivity on the file name to retrieve the correct file
    - Display the result upon successful operation
    - Display the result upon unsuccessful operation
  + Option to navigate back to the main context
  + There should be a third option to close the application
  + Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance
* **You must use the following:**
* Eclipse/IntelliJ: An IDE to code for the application
* Java: A programming language to develop the prototype
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Search and Sort techniques: Data structures used for the project
* Specification document: Any open-source document or Google Docs

**Following requirements should be met:**

* The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
* Document the step-by-step process starting from sprint planning to the product release.
* Application should not close, exit, or throw an exception if the user specifies an invalid input.
* You need to submit the final specification document which includes:
  + Project and developer details
  + Sprints planned and the tasks achieved in them
  + Algorithms and flowcharts of the application
  + Core concepts used in the project
  + Links to the GitHub repository to verify the project completion
  + Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

4. USER STORY

Company Lockers Private Limited has hired a full stack developer to digitize their product, they have the aim to develop an prototype of application where a user can add, search & delete a file for business level operations. The application should contain the name of the first product LockedMe.com

The application should also be able to sort the file name in ascending order & it should contain the welcome screen for their application LockMe.Com. The prototype should contain all the features as prescribed by the organization .The application source code should be uploaded to github repository.

1. SPRINTS & PLANNING

|  |  |  |
| --- | --- | --- |
| **SR NO** | **ACTIVITY** | **TIME REQUIRED** |
| 1. | SPRINTS | 7 DAYS |
| 2. | Account setup on GITHUB | 1 HOUR |
| 2. | Installing GIT Desktop | 1 HOUR |
| 3. | Installing Eclipse IDE for Windows | 1 HOUR |
| 4. | Coding in Java | 7 DAYS |
| 5. | Push the code in GIT HUB Repository | 1 HOUR |

1. The sprint activity includes all major aspect of developing a project, from developing a user story to planning day to day activity which includes coding, testing & delivery of the final project to the user.
2. The sprints cover all the activity & how much time is required to develop the entire project.
3. The work is being divided with each & every member of the team and daily progress chart is being prepared to check the progress of the entire project.
4. In the first phase of the code development we do the following activity:

* For this project first we need to create an account on github.
* This is the second major activity is to install the GIT on Desktop.
* Third and most important step is to install an IDE for Java.
* This will help you to run and execute the code. We can use either IntelliJ or Eclipse IDE.
* Here in our project I have used Eclipse IDE for Windows.
* The fourth step is to start coding on the project in JAVA IDE.
* The last and most important step is to push the code in the Github repository.

5. AIM

* The aim of the project is to develop a program which contains the file operations.
* The major file operation is to add, search & delete a file.
* This project will help to create an application using oops concepts
* This purpose of this project to develop the prototype of the application

6. OBJECTIVE

* The objective of the project is to use the concepts of java to create the file handling operations.
* To understand and implement the core java concepts for project development.
* To add the business level operation such as insert, display, insert & delete operations.
* We have to create a menu drive program by using the do –while loop which will consist of all the operation.

7. SYSTEM REQUIREMENTS

* **System Requirement for Eclipse IDE (For Microsoft Windows)**

|  |  |  |  |
| --- | --- | --- | --- |
| SR NO | REQUIREMENT | MINIMUM | RECOMMENDED |
| 1 | JAVA VERSION | 1.4.0 | 5 or Greater |
| 2 | MEMORY | 512 MB | 1 GB or Greater |
| 3 | Free Disk Space | 300 MB | 1 GB or More |
| 4 | Processor Speed | 800 MHz | 1.5 GHz |
| 5 | Windows OS | Windows Xp | Windows 7 or Greater |

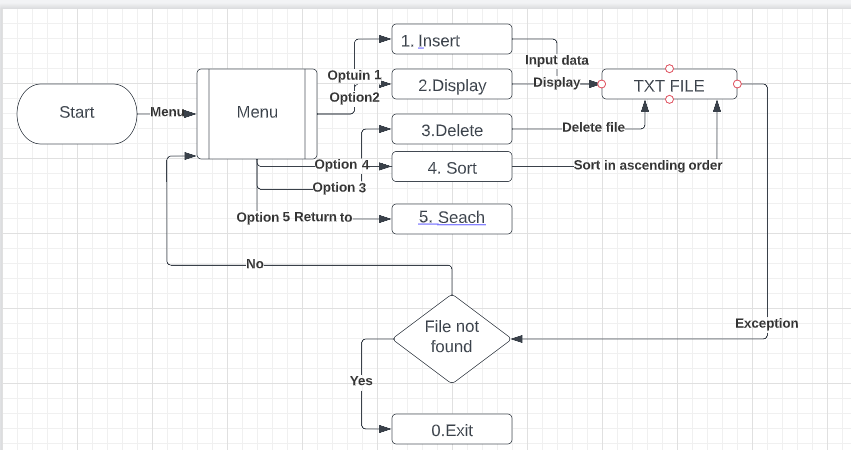
* **System Requirement for GIT Desktop**
* OS: Windows 7 or Greater( 32 bit /64 bit)
* Memory: 1 GB OR Greater
  1. DATA FLOW DIAGRAM
* DFD LEVEL 0 DIAGRAM

OUTPUT

USER

* This is data flow diagram of level zero.
* There are three block in this diagram the user will input the data & the process block will execute the code.
* The final output will be obtained in Eclipse IDE.

FLOWCHART



* The flow chart has seven process block, one decision block & one predefine process block & start block.
* The start block begins with the compilation & execution of the program.
* The Menu block or the predefine block will give the user to select one option from the menu.
* The process blocks represent the operations such as insert, sort, display, delete & search the file.
* The decision block represents the exception condition.
* The exit block will return the function to the main directory.
  1. PROPOSED METHOD

1. The project will adopt the following components while implementing the methodology to achieve the expected results:
2. We will use the do-while loop condition in our program.
3. We will use the collection & array as well.
4. We will have to use try and catch exception method otherwise during runtime we will get the error.
5. We have to write the program in Java IDE i.e. Eclipse.
6. This will help us to identify the error during the coding such as syntax error.
   1. JAVA CODE WITH OUTPUT

**package** Lockmecom;

**import** java.util.\*;

**import** java.io.\*;

**class** Lockmecom **implements** Serializable{

**int** pno;

String pname;

**int** price;

Lockmecom(**int** pno, String pname, **int** price){

**this**.pno = pno;

**this**.pname = pname;

**this**.price = price;

}

**public** String toString(){

**return** pno+" "+pname+" "+price;

}

}

**class** LockmecomDemo{

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

**int** choice = -1;

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

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

File file = **new** File("lockmecom.txt");

ArrayList<Lockmecom> al = **new** ArrayList<Lockmecom>();

ObjectOutputStream oos = **null**;

ObjectInputStream ois = **null**;

ListIterator li = **null**;

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

}

**do**{

System.***out***.println("LOCKME.COM MENU OPTIONS");

System.***out***.println("1.INSERT");

System.***out***.println("2.DISPLAY");

System.***out***.println("3.SEARCH");

System.***out***.println("4.DELETE");

System.***out***.println("5.UPDATE");

System.***out***.println("6.SORT By PNo - On Screen");

System.***out***.println("7.SORT By PNo - In File");

System.***out***.println("8.SORT By PName - On Screen");

System.***out***.println("9.SORT By PName - In File");

System.***out***.println("10.SORT By Price - Desc - On Screen");

System.***out***.println("11.SORT By Price Asc - In File");

System.***out***.println("0.EXIT");

System.***out***.print("Enter Your Choice : ");

choice = s.nextInt();

**switch**(choice){

**case** 1:

System.***out***.println("Enter how many product you want : ");

**int** n = s.nextInt();

**for**(**int** i=0;i<n;i++){

System.***out***.print("Enter Product No: ");

**int** pno = s.nextInt();

System.***out***.print("Enter Product Name: ");

String pname = s1.nextLine();

System.***out***.print("Enter Product Price : ");

**int** price = s.nextInt();

al.add(**new** Lockmecom(pno,pname,price));

}

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

**break**;

**case** 2:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 3:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

**boolean** found = **false**;

System.***out***.println("Enter pno to Search : ");

**int** pno = s.nextInt();

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

li = al.listIterator();

**while**(li.hasNext()){

Lockmecom e = (Lockmecom)li.next();

**if**(e.pno == pno){

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

found = **true**;

}

}

**if**(!found)

System.***out***.println("Record Not Found...!");

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 4:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

**boolean** found = **false**;

System.***out***.print("Enter pno to Delete : ");

**int** pno = s.nextInt();

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

li = al.listIterator();

**while**(li.hasNext()){

Lockmecom e = (Lockmecom)li.next();

**if**(e.pno == pno){

li.remove();

found = **true**;

}

}

**if**(found){

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

System.***out***.println("Record Deleted Successfully....!");

}

**else**{

System.***out***.println("Record Not Found...!");

}

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 5:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

**boolean** found = **false**;

System.***out***.print("Enter pno to Update : ");

**int** pno = s.nextInt();

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

li = al.listIterator();

**while**(li.hasNext()){

Lockmecom e = (Lockmecom)li.next();

**if**(e.pno == pno){

System.***out***.print("Enter New Product Name : ");

String pname = s1.nextLine();

System.***out***.print("Enter new Price : ");

**int** price = s.nextInt();

li.set(**new** Lockmecom(pno,pname,price));

found = **true**;

}

}

**if**(found){

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

System.***out***.println("Record Updated Successfully....!");

}

**else**{

System.***out***.println("Record Not Found...!");

}

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 6:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e1.pno - e2.pno;

}

});

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 7:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e1.pno - e2.pno;

}

});

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 8:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e1.pname.compareTo(e2.pname);

}

});

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 9:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e1.pname.compareTo(e2.pname);

}

});

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 10:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e2.price - e1.price;

}

});

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

**case** 11:

**if**(file.isFile()){

ois = **new** ObjectInputStream(**new** FileInputStream(file));

al = (ArrayList<Lockmecom>)ois.readObject();

ois.close();

Collections.*sort*(al, **new** Comparator<Lockmecom>(){

**public** **int** compare(Lockmecom e1, Lockmecom e2){

**return** e1.price - e2.price;

}

});

oos = **new** ObjectOutputStream(**new** FileOutputStream(file));

oos.writeObject(al);

oos.close();

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

li = al.listIterator();

**while**(li.hasNext())

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

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

}**else**{

System.***out***.println("File not Exists....!");

}

**break**;

}

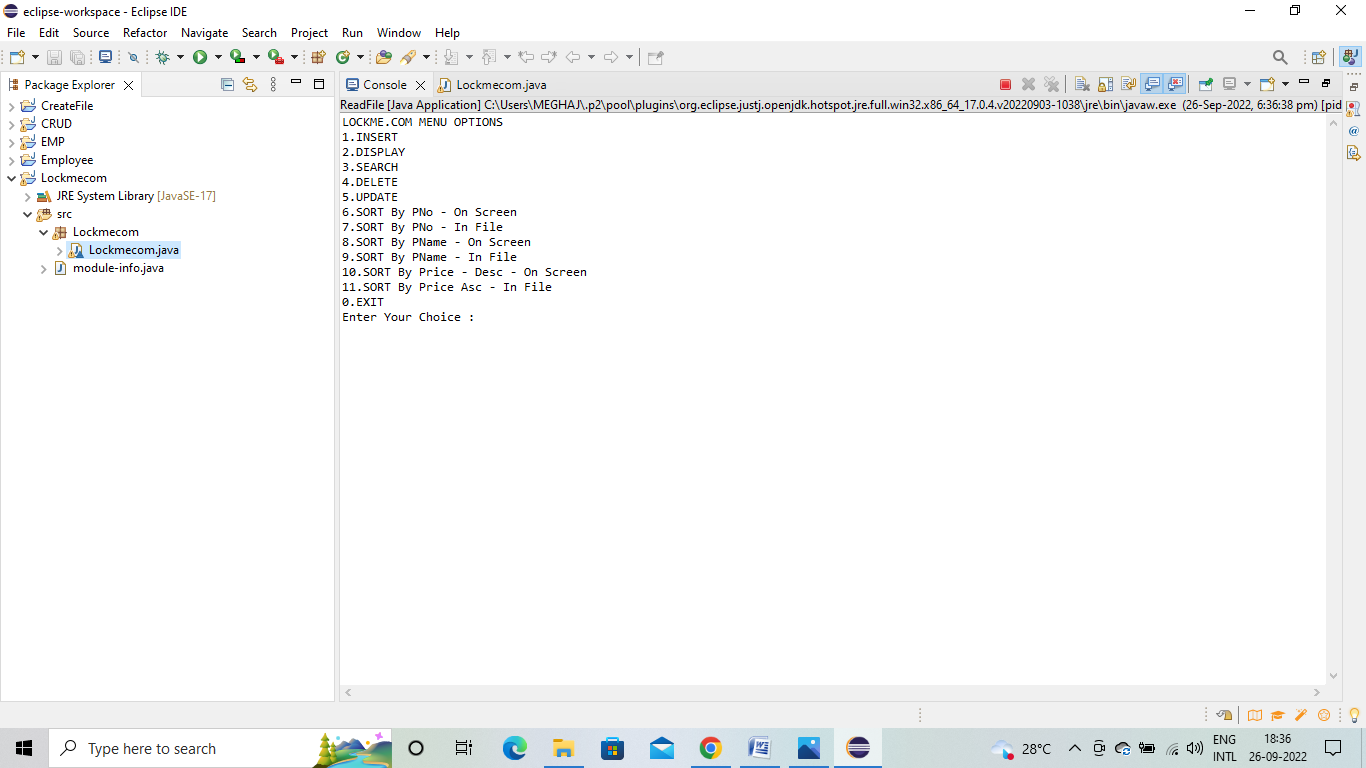
}**while**(choice!=0);

}

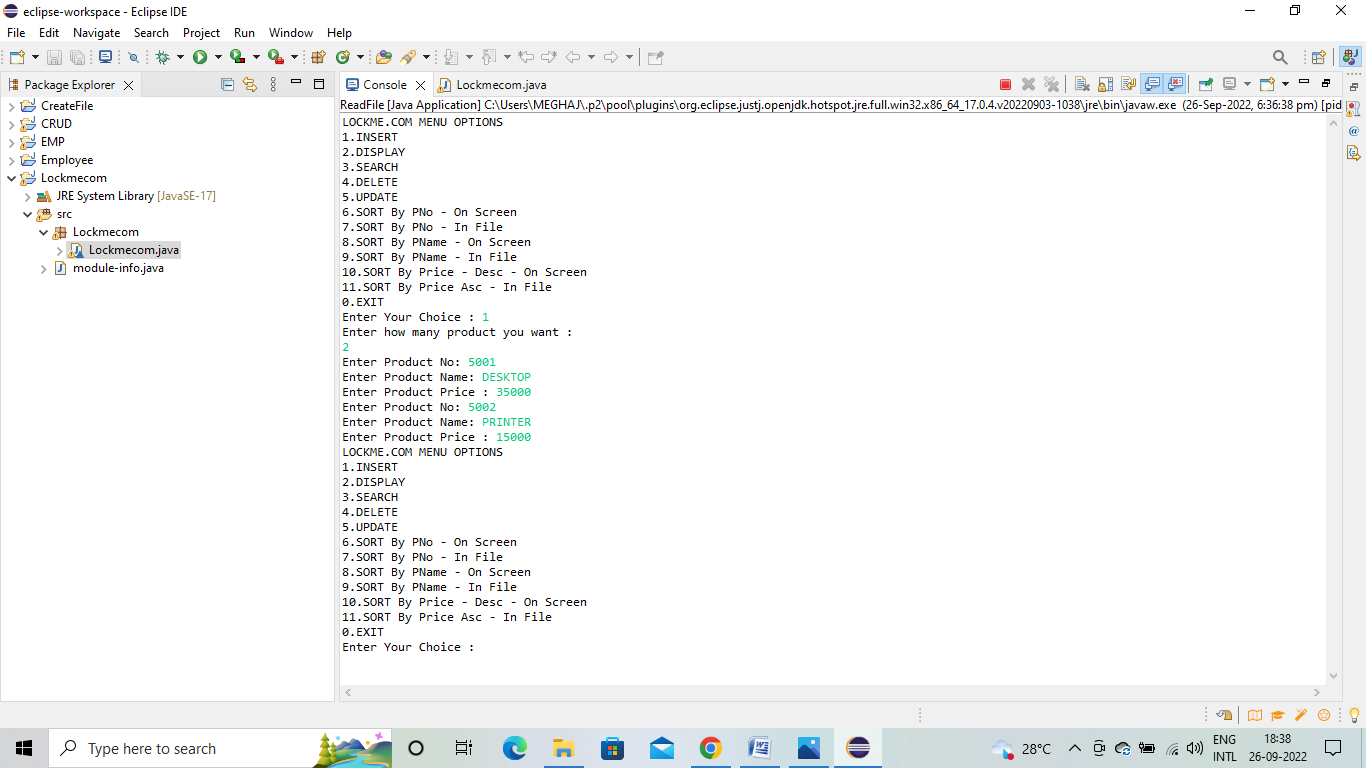
}

OUTPUT OF THE PROGRAM

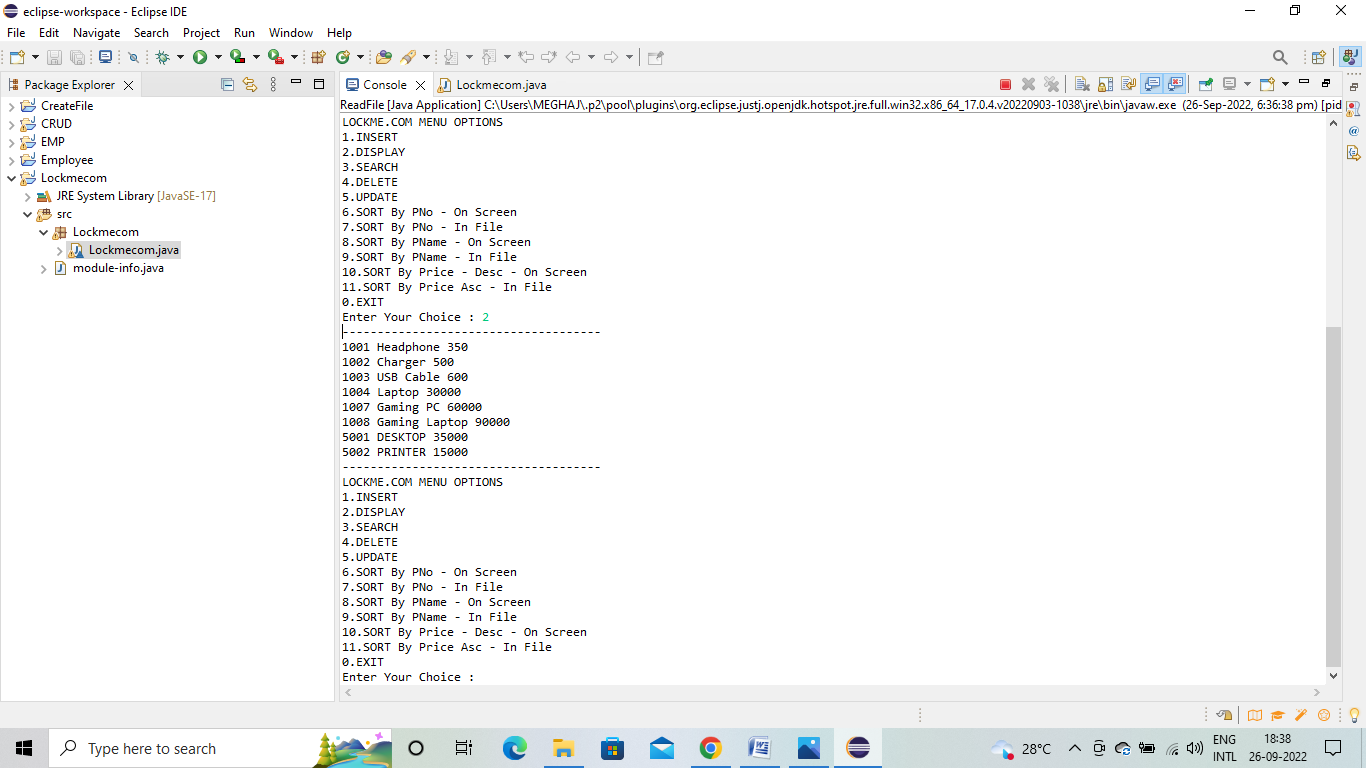
1.MAIN MENU OPTION SCREENSHORT



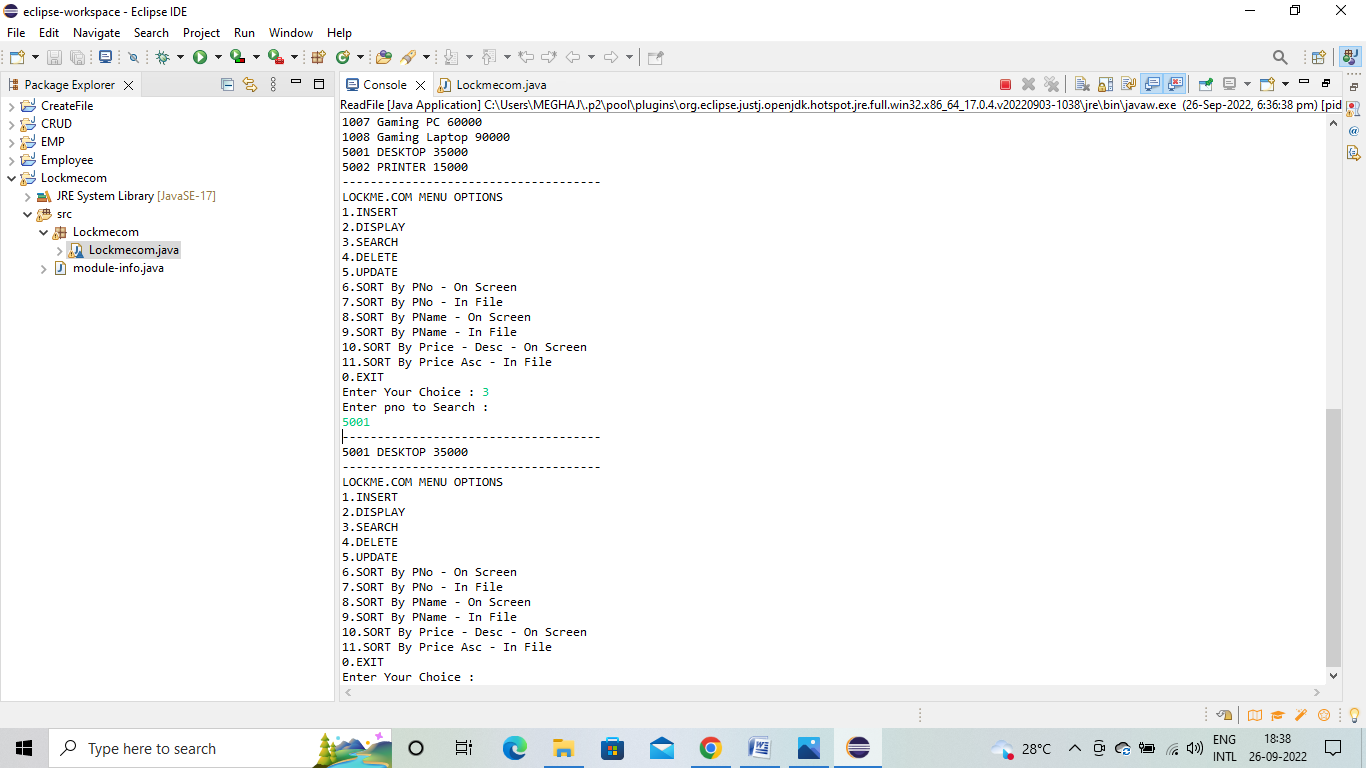
2. INSERT OPERTION SCREENSHORT



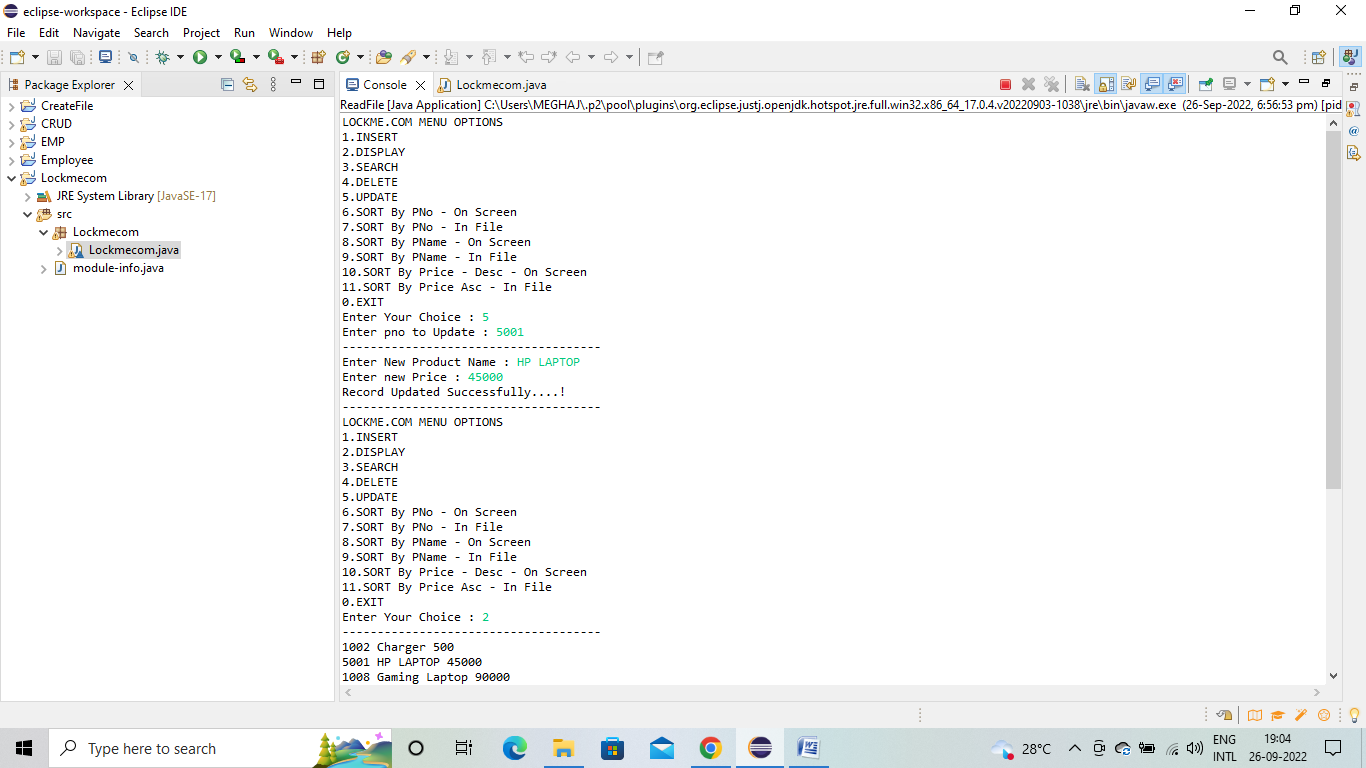
3. DISPLAY OPTION SCREENSHORT



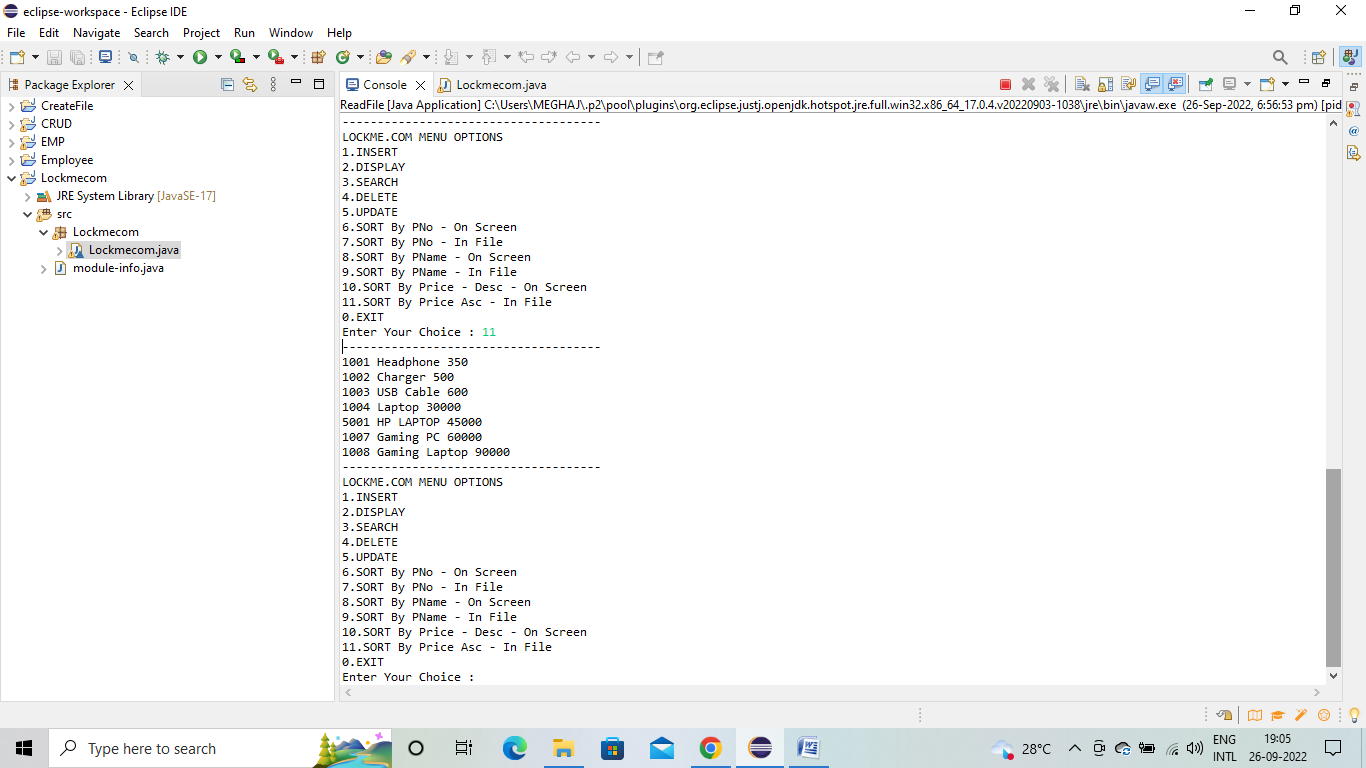
4.SEARCH OPTION SCREENSHORT



5.UPDATE OPTION SCREENSHORT



6.SORT OPTION SCREENSHORT



* 1. REFERNCE

1. <https://lms.simplilearn.com/courses/3850/Implement-OOPS-using-JAVA-with-Data-Structures-and-Beyond/syllabus>
2. <https://www.geeksforgeeks.org/file-handling-in-java/>
3. <https://www.w3schools.com/java/java_files.asp>