

**PROGRAMMING TECHNIQUES**

Homework 2

**Student:Nicolescu Alexandru**

**Group:** 30422

**Table of Contents**

**1. Objective 3**

**2. Problem analysis, modeling, scenarios, use cases 3**

**2.1. Problem Analysis 3**

**2.2. Modeling 4**

**2.3. Scenario and use cases 4**

**3. Design 5**

**3.1. UML Diagram 5**

**3.2. Data Structures 6**

**3.3. Class Projection 6**

**3.4. User Interface 8**

**4. Implementation and use of cases 9**

**5. Results 10**

**6. Conclusion and possible updates 10**

**7. References 10**

**Documentation OrderManagement for processing customer orders**

1. Objective

This homework’s aim is to use elements of object-oriented programming to implement a data management application which uses MySQL Workbench in order to store the data efficiently.

1. Problem analysis, scenarios, use cases

*Consider an application OrderManagement for processing customer orders. The application uses (minimally) the following classes: Order, OPDept (Order Processing Department), Customer, Product, and Warehouse. The classes OPDept and Warehouse use a BinarySearchTree for storing orders.*

*a. Analyze the application domain, determine the structure and behavior of its classes, identify use cases.*

*b. Generate use case diagrams, an extended UML class diagram, two sequence diagrams and an activity diagram.*

*c. Implement and test the application classes. Use javadoc for documenting the classes.*

*d. Design, write and test a Java program for order management using the classes designed at question c). The program should include a set of utility operations such as under-stock, over-stock, totals, filters, etc.*

2.1 Problem analysis

If we take a closer look to the problem of order management we shall find that it is a more complex problem that it may seem at first glance.

First of all we need to find an OOP style way to store the data that comes from the MySQL Workbench. I have used an object for each type of table in MySQL. Therefore we have in the package model: Order(to take the orders from MySQL), Client, Product and the OrderDisplay(to store the Product name, while Order only stores its ID, Client name and so on).

Another problem concerning the order processing would be the way the user gives us the data and the way we tell him the result of the processing. After doing some research on the internet and looking at online applications that provide these services that cover the order processing problem I have reached the conclusion that the best way in terms of displaying data which comes from tables is using naturally a JTable which is put in the Graphic User Interface along with the data input section.

Further on we shall analyze all the aspects which needed to be managed in order for the right functioning of this project.

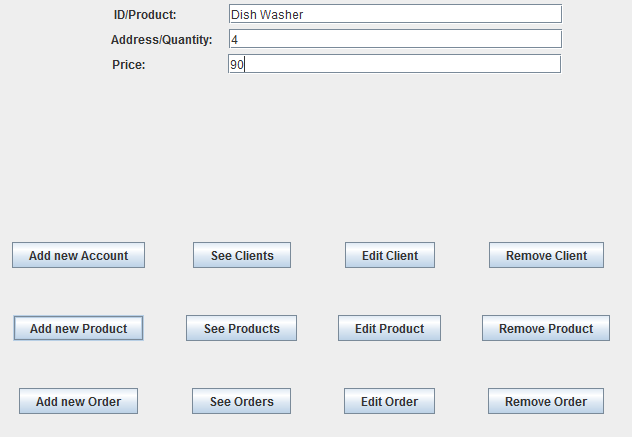
* 1. Modeling

The implementation of this project is the same one as the one presented in the section 2.1. The “BLL” package which stands for business logic layer has all the rules of storing and displaying data while the “GUI” package naturally only makes the application-user connection using a very smooth and efficient data inputting. Finally, the DAL which stands for data access layer is used only for CRUD operations which are: Create, Read, Update, Delete. There operations are implemented for each of the 3 important Models: Order, Client, Product.

2.3 Scenarios and use cases

The scenarios were already mentioned, but I will present the details here. Firstly I thought about how the data will be managed by the application and my idea was: GUI for reading the data -> BLL for verifying the integrity of the data -> DAL to store or request the data from MySQL -> BLL for further processing the data -> GUI for displaying. I followed this plan and I did not have any unexpected surprise when it comes to the implementation and the structure of the program.

The use cases are strictly dependent on the user, and finally I order to make the application as user friendly as possible I decided to implement the following user interface:



From now on I will present the functionality of the application using the Product functions as the one of Client and Order processing are very similar.

-Create: The user needs to input the name, quantity and price of the new product and then press Add new Product button which creates and the new list of products.

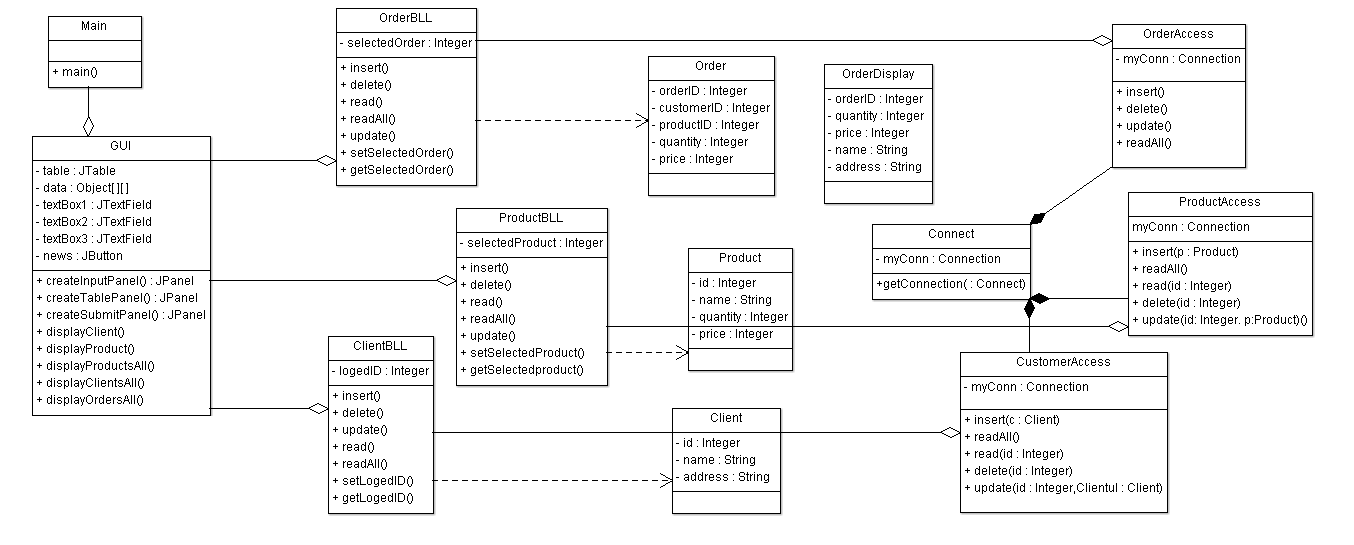
-Read: See Products displays all the current products no matter they are or no on stock.

-Update: The user needs to input the Product name, quantity or price or any 2 of these or even all of the data that needs to be edited, select the product to be edited in the table and press Edit Product.

Delete: Select the product to be deleted in the JTable and just press Delete product.

Similarly, the rest of the operations are done by this pattern.

1. Design

3.1 UML Diagram

The UML diagram is a class diagram in which we can find the relationship between classes and also the elements that the specified class contains.

One could observe that for joining the classes I used several types of relationships. Between the class GUI and the BLL classes I used “uses” relationships, between BLL and DAL classes there is also a uses relationship. The difference is that DAL classes have a composition relationship because Connect class could not exist without DAL classes. The objects formed by Model classes are used for sending data between DAL, BLL and GUI packages.

3.2 Data structures

In this application I have used various data types including JButtons, JTextFields, JPanels, JFrame for the GUI package, Client, Product, Order for transferring data between packages and the classic int, String, Boolean for regular operations.

3.3 Class projection

Class projection refers mainly to how the model was thought, how the problem was divided in sub-problems, each sub-problem representing more or less the introduction of a new class. First I will start by mentioning exactly how my problem was divided into packages and afterwards each package with its own classes. I begin by creating the three packages I used: the first one being called “gui” and the second one being called “bll”, third “data.access” and the “main” package that only starts the execution. I named them intuitively because the first one handles the interface; the part that deals with the user and the second one handles the implementation, the part that is hidden from the user and the third one makes the connection to the database.

1. *gui* package

This package only contains the GUI class which creates the User Interface and moreover it is used for input-output functions. Using JTextField it takes the input from the user and sends it to processing. Using the JTable it displays the results of the operations performed. The GUI class is also used to enable the user to choose what operations he wants to be done.

**this**.setLayout(**new** GridLayout(1, 2));

**this**.setSize(1300, 700);

JPanel output = createTablePanel();

JPanel inputPerson = createInputPerson();

**this**.add(inputPerson);

**this**.add(output);

**this**.setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

**this**.setVisible(**true**);. . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . .

This is the main code of the GUI class which describes perfectly how everything is done. There are 4 methods used to create the main Frame, one for each Panel which is added to the window.

2) bllpackage

This package contains all the classes which do the CRUD (create, remove, update, delete) for each Order, Client and Product.

It contains one class for each type of object and all these classes follow a certain pattern found in the implementation of all ClientBLL, OrderBLL, PackageBLL.

**public** OrderDisplay[] read()

**public** **boolean** update(Order order)

**public** **boolean** delete()

**public** **boolean** insert(Order order)

**public** **int** getSelectedOrder()

**public** **void** setSelectedOrder(**int** selectedOrder)

This implementation varies from object to object. A good example of how efficient is the usage of the bll is the data control in customer operations. We shall analyze their code:

**public** **boolean** update(Client clientul) {

**if** (logedID != 0 && ((clientul.getName()!=**null** && !clientul.getName().isEmpty()) || (clientul.getAddress()!=**null** && !clientul.getAddress().isEmpty()))) {

database.update(clientul,logedID);

**return** **true**;

} **else**

**return** **false**;

}

**public** **boolean** insert(Client client) {

**if** (client.getName() == **null** || client.getAddress() == **null** || client.getName().isEmpty()|| client.getAddress().isEmpty()) {

**return** **false**;

} **else** {

database.insert(client);

**return** **true**;

}

}

**public** **boolean** delete() {

Client clientu = database.read(logedID);

**if** (clientu.getId() != 0)

database.delete(logedID);

**else**

System.***out***.println("Invalid ID");

**return** **true**;

}

In this problem of inserting data I verified if the inputted data is correct in terms of the name type, address type and for example when entering a product I have verified that the quantity and price are positive numbers.

As another example of how user friendly and efficient the application is all the errors of the BLL, which checks for logical problems in the inputted data, are transmitted to a NEWS button on the user interface which displays certain messages. We shall see this at 3.4.

3) data.access package

This package is used mostly for reading data from the MySQL Workbench and this is done by a Connection which is defined in the Connect class and is returned to each of the DAL classes using a static function *getConnection().* We shall illustrate snippets of code for you to get how the connection is done:

**public** **void** insert(Client c) {

**try** {

String nume = c.getName();

String adresa = c.getAddress();

myConn = Connect.*getConnection*();

Statement myStat = myConn.createStatement();

String sql = "insert into customers" + "(name, address)" + "values ('" + nume + "','" + adresa + "')";

myStat.executeUpdate(sql);

} **catch** (Exception exc) {

exc.printStackTrace();

}**finally** {

**if** (myConn != **null**)

**try** {

myConn.close();

} **catch** (SQLException e) {

e.printStackTrace();

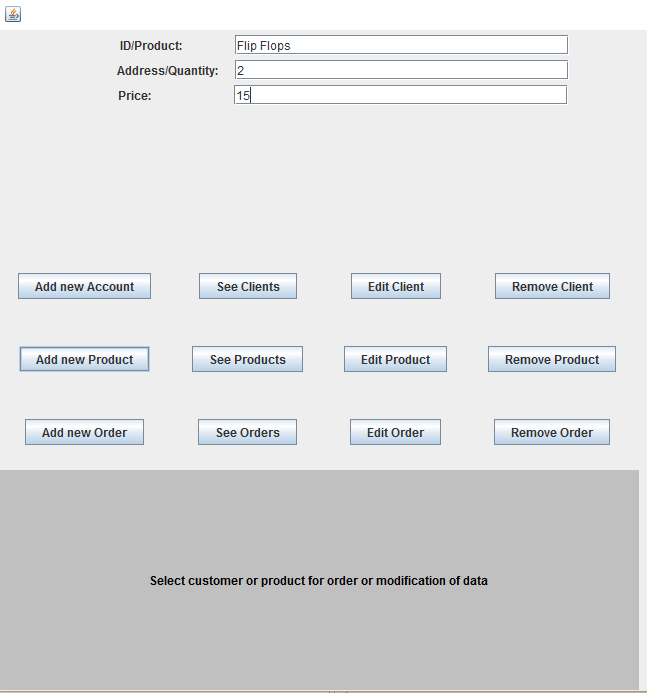
}

}

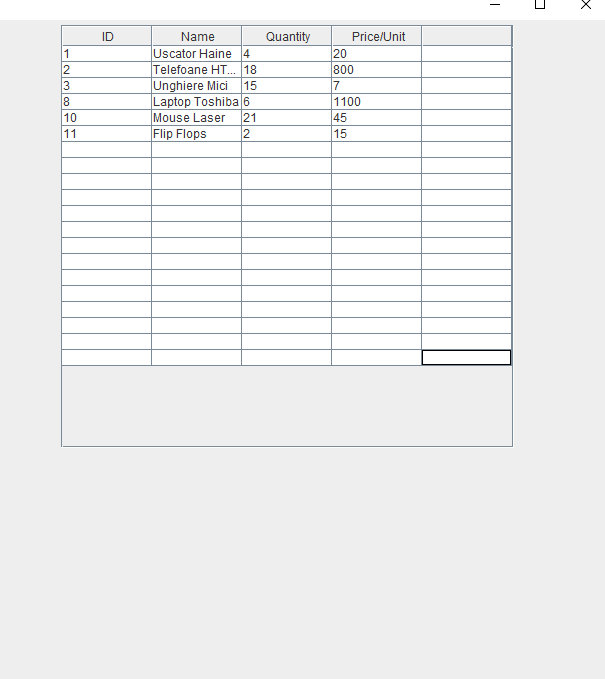
}

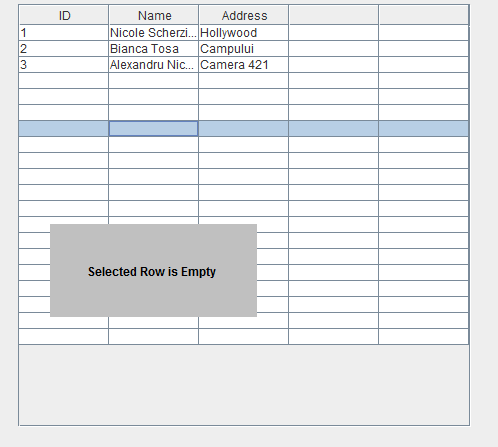
3.4 The interface

The interface is made out of 2 JPanels which are included in the JFrame in a GridLayout(1,2). We shall describe each JPanel in order to understand how the user interface is created.



This photo represents the input panel which is used in order for the user to enter the command data and/or select the command. Some of the data can be selected from the table. There is also a news button which is located on the button of the panel and it displays all the information for using this application and if the case, the errors resulted out of entering the wrong kind of data, therefore the use of this application is easy and does not need any other extra information.





This second photo shows how the resulted queries or edits of the data are displayed to the user using a very straight forward table that displays only the essential information. Numerous bugs are treated in this project such as: selecting no item and trying to deleting it, trying to add a number of “char” products and so on.

1. Implementation

In what the implementation is concerned this project was developed in Eclipse and it was only tested in this environment. However the program should maintain its portability. Concerning the code implementation I did not make use of laborious algorithms, but I have rather stayed faithful to the classical algorithms of computing polynomials learned in high school. However I have tried to implement my problem in a way that appears to me as being the most efficient one, this is why I have changed my model at first. Testing implies checking for any errors in the program or limitations of this program. Due to the fact that the program is rather simplistic, they are few errors that might generate this program to work wrong or to stop. These errors are mostly related to the interface or the database connection. I have assumed that the user reads the instructions from the interface and respects them, otherwise if he enters data with invalid format the program will probably generate some bugs and will stop. Hence this part with checking all the possible scenarios will be seen as future development.

1. Results

The application is an user friendly and useful application to perform basic create update delete and read operations on a sample created database. As the application is developed on a Java platform, it is highly portable and allows it to run on several operating systems (as long as they have the Java SDK installed). The application is straightforward an easy to understand and to use by any user who respects the instructions given in the interface and who has some basic knowledge of database storing, of course. Even though being limited, this application can be considered as being a helpful tool that can be used when dealing with such data storing situations.

1. Conclusions

All in all, the application works perfectly on the required operations and it is a user friendly interface which gives the users a simple and efficient answer to any problems related to a selected database operation. As in possible updates I would add more fields to the database and have specific constraints implemented in the BLL package, but this of course depends on the application that this program will have.

1. References
2. <http://stackoverflow.com/questions/20327005/jtable-actionlistener-for-select-a-row>
3. https://www.youtube.com/watch?v=2i4t-SL1VsU