# Assignment 3 Documentation

# Order Management

Dreghici Popa Vlad

Group 30422

1. *Problem description*

Consider an application OrderManagement for processing customer orders for a warehouse. Relational databases are used to store the products, the clients and the orders.

1. *Problem analysis*

Order management system are usually used in large online shops like emag, cell pc garage and others. They are conveniently made in such a away that a user can create an account and after that view al the items in the shop, review their specifications, look for a particular product and after they have decided on the purchase, they can create an order. Also, those system are easy to use by the owners, being less of an effort to analyze placed orders, user and products.

The project should be able to connect to a database which contains three tables, one for users, one for products and one for orders. This database is made using a software called Mysql Database which simulates a server of tables on a localhost. The fields of the user table should be id, name, email, addres or phone number. The fields of the product table are id, name, quantity left in stock and finally the fields of order are id, idcustomer, idproduct, customer name, product name and quantity requested.

The user of the application should be able to input the following data: realize operations from the gui directly in the database, operations for each table are Show all, update, delete, insert and other necessary ones in order to get the data from tables.

The output is shown in a JTable. The table is created for each table from the database, resulting in the end we have 3 tables. One for customers, one for products and one for orders. For each table the user will be able to see all the data in it.

Run application

User

Go back

Review Orders

Select option

Place order

View Data

1. *Design*

*Classes*

The classes is a blueprint that describes the behavior of the object it supports. The classes are organized in packages, which basically are a collection of classes, interfaces. In our case, the classes are organized in a MVC manner.

An object instantiated from a class has states and behaviors declared in the respective class.

The project will be split in a certain manner. There will be 4 packages: Model, Business Logic ( which is the controller ), Presentation (gui classes) and finally Data Access which contains all the classes that execute operations on the database we access. The model package will contain all the back end processes, like the algorithms, client, orders, products. The view package will contain the front end classes. In this case we have for classes: Start, Clients Page, Products Page and Orders page. This is the visualization of our application.

**Client class:** will implement the attributes of a client from the database. All the attributes should be named exactly as named in the database. It will have as attributes and id, name, email and address. Each client will have an unique id in order to be identified when making a order.

**Product class:** In our case, the product class will be similar to the client class. It will also contain all the attributes from the data base: id, name, price and quantity available and will have the getters and setters methods implemented.

**Orders class:** this class is the intermediate between the products and users. It will contain an unique order id and also the id of the customer and product. Also it will hold the name of the product and the customer and the quantity the user ordered.

**Controller class:** the controller class is the main class of the app. It basically takes all the input given by the operator and inserts it in our algorithm. The controller class also makes the link between the database connection and the application gui, it will manage all the interactions of the user with the app.

**Gui classes:** those classes are similar between them. All of them implement a JTable which displays the data from the database, and for each case we have certain operations. For clients and products we have the options to delete, update, insert and get all data from data base and for orders we have the options to insert a new order and to delete an order.

**DATA ACCES LAYER:** Those classes are the links between the application and the data base. First of all we have a database connection class

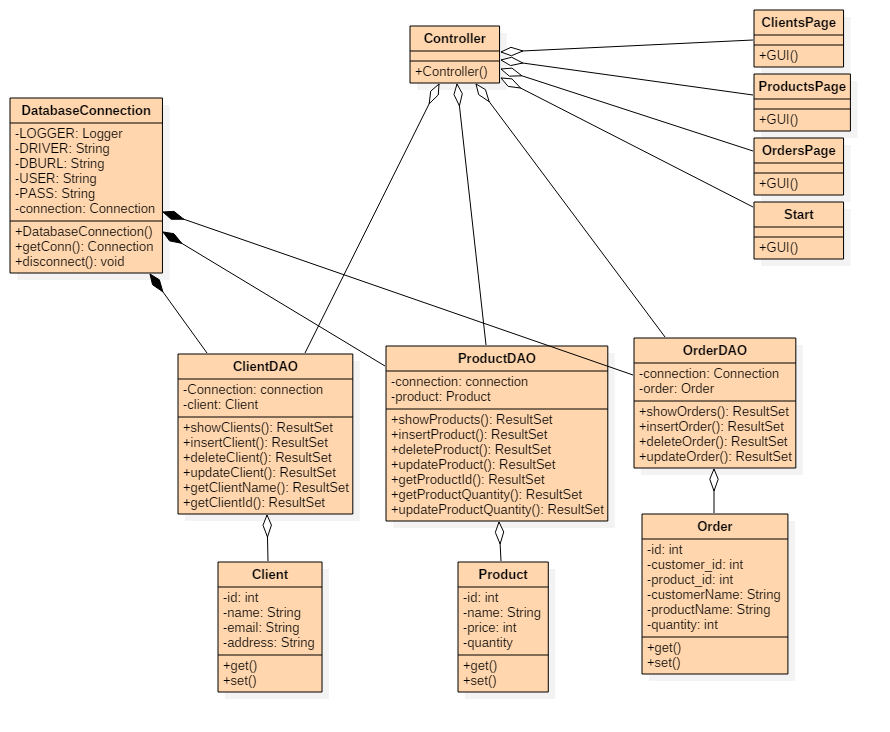
**DatabaseConnection Class:** this class establishes the connection between the app and the data base we created. In here we have declared all the needed attributes in order to access the database.

**DAO Classes:** those classes manage all the queries that need to be done on a certain table. They will contain the basic queries needed to be executed on each table like: SELECT, INSERT, UPDATE, DELETE

**ClientDAO class:** client dao class gets a connection from the database and has a number of ResultSet methods which contain queries that need to be executed on the clients table. It will also contain certain methods to ease our work with the clients and creating the table in the GUI

**OrderDAO Class:** similar to the clientdao class, contains all the queries to be executed in the table.

**ProductDAO Class:** this class is also similar to the other two classes. It will additionally contain some extra queries like select ID and Select Name to help use when needed to introduce the product in the order and will also contain a method which will increment the quantity of the product when needed.



1. *Implementation*
   1. *Client Class*

In real life, a customer usually has a few basic states and behaviors. Each customer is unique, this uniqueness will be given by the id. A client has a name. In our case, the name will be implemented as an name. Also, each customer has way to contact him which is the email and a way to receive orders, the address. All those attributes will have getters and setters in order to be more easily accessible.

* 1. *Product Class*

Similar to the client class, the product class will contain certain attributes each product should have: id for uniqueness, a name, a price and the quantity available in stock. Again all those attributes will have getters and setters

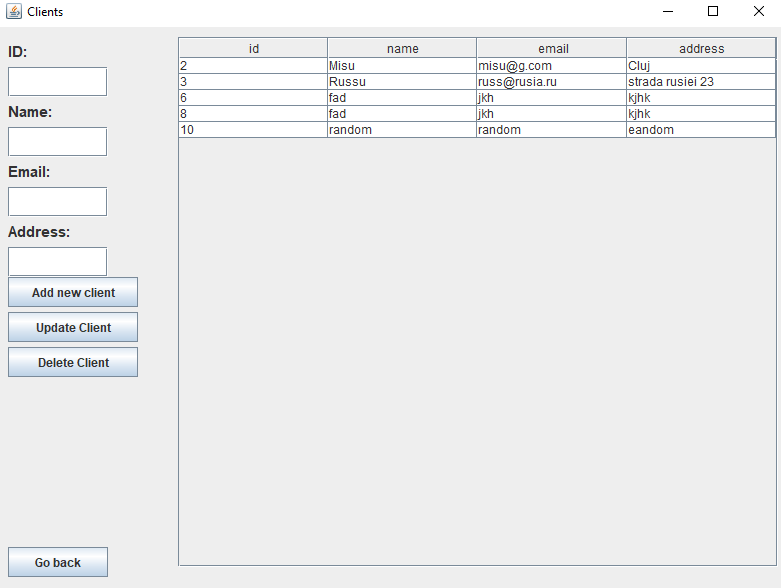
* 1. *Order Class*

Being another table in the data base, this class will also contain the attributes from the table, those being the order id, the ids of the clients and products requested, the quantity of the order and the names of the client and products.

* 1. *Controller Class*

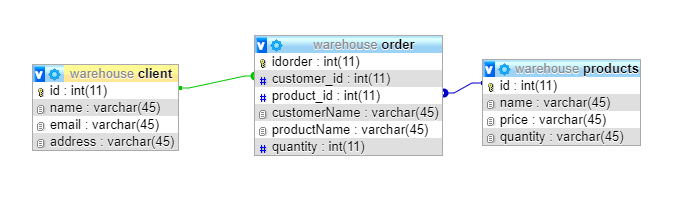
The controller class is exactly what the name says, a controller. It will implement the business logic of our application. The controller will also link the presentation part with the data access layer. First of all it will connect to the database. It will execute certain queries one a particular button is pressed. For example in the clients menu when the admin wants to insert a new client, the admin will have to complete the necessary fields with the data of the client and after that press the add new client button. Once this button is pressed the controller will make the link between the gui and the database, perform the requested query and insert the client in the table. This is the procedure for most of the algorithms implemented.

* 1. *Gui Classes*



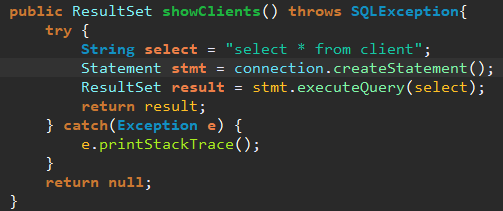
The gui classes are the front end of our applications. They will display all the data taken from the tables and will make the interface user friendly. The gui is separated in 3 parts. The clients page which contains a table with data of the clients from the database, 4 labels: id, name, email, address and 4 textfields where the data is inputed and 3 buttons to perform certain queries on the table. Similar to the clients page, the products page also contain a table with products from the database, with the following elements: a table which contains all the data from the data base, 4 labels: id, name, price and quantity and 3 buttons, each for executing a query on the table. Finally the orders page contains a table with the placed orders in the database, 2 combo boxes to select the client and the product to be added, a text field where to place the quantity and an id to delete the an order.

* 1. *DatabaseConnection class*

This class is basically the one that executes the connection between the app and the database. It contains a logger, a driver for the database, a url of the database and the user and pass that are needed to access the data. In the constructor we get the driver and create the connection to the database. After we call the constructor we are connected to the database and in order to disconnect from it we can call the method disconnect which will end the connection. 

* 1. *ClientDAO class*

Now for each table from the database we have a DAO class which is basically a data access object. Those classes will be able to connect to the database and execute the requested queries from the GUI. In this particular case the ClientDAO class contains a constructor in which we will get the connection from the database connection. After the constructor we have a number of queries each executing a certain task. The following is an example of the select all query:



This query executes a select all from clients. This will be used to fill the table in the GUI.

Other queries in the client class are: INSERT, which is used to add a new user in the database along with all the attributes of it, DELETE, used to delete a certain client after a given ID, UPDATE, used to modify the data of a client by changing the fields, get clients name and get clients id used to place orders in the OrderDAO class.

* 1. *ProductDAO class*

The product dao class is similar to toe clientDao class, contains the logic and the querries for the products in the database. In this class we have a SELECT ALL method, a INSERT method, DELETE and UPDATE. Also to ease our work there are 4 additional methods to get the name, id, quantity and update quantity of a product. Those methods are mostly used in the GUI where we have to create the order and select the name of a product from the combo box.

* 1. *OrderDAO class*

The orderDao class is also similar to the other DAO classes. It contains all the queries necessary to work with an order. There is no point in adding details here, everything is similar to the other DAO classes.

1. *Testing*

The testing part will be done by repeatedly using the application, inserting, deleting, updating clients, products and orders. In order to debug much easier, the details will be printed in the console of the eclipse IDE.

1. *Results*

During the usage of the application the user will be able to execute certain queries on the three tables from the database. First of them is the client table, here the user will be able to insert a new client, update an existing client or even delete a client by inserting the necessary info. Second, the product table, where the user will be able to execute similar commands on the products as on the clients. Lastly, the orders table, where the user will be able to place an order by selecting a client, a product and inserting a quantity number. Also in the order control, the user will be able to delete an order if necessary

1. *Conclusions*

By working on this assignment I acquired more knowledge about how to work with a database, creating connection by using the Apache and MySql servers. Also I learnt how to write queries in java as Result Sets and connect them to the database. Another important lesson from this homework is the reflection technique. The reflection technique helped me to construct the tables needed to display the data from the database

As a future feature of this application I thought of implementing another table which could be used in order for the user to be able to create an order with more products than just one, and for each product there would be a quantity number. Another future feature would be including another table for the producer of products and maybe one for warehouses where they are located. This would help the customers to know from where to order the products so that they arrive faster.

1. *Bibliography*

<http://stackoverflow.com/>

<http://www.caveofprogramming.com/>

<https://tutorialspoint.com/>

<https://www.javatpoint.com/>

<https://geeksforgeeks.com/>