**Assignment 3**

**Programming Techniques**

Order Warehouse Application

**Student:**Boian Maria Andreea

**Group**:30423

Teacher: Ioan Salomie

Lab assistant: Claudia Pop

2 nd semester 2017-2018

1.**Assignment objective**

**1. 1. Task description**

Create a graphical interface with a window for client operations: add new client, edit client, delete client, view all clients in a table (JTable), a Window for product operations: add new product, edit product, delete product, view all product in a table (JTable). Create a product order for a client: the application user will be able to select an existing product, select an existing client, and insert a desired quantity for the product to create a valid order. In case that there are not enough products, an under stock message will be displayed. After the order is finalized, the product stock is decremented. Use reflection techniques to create a method that receives a list of objects and generates the header of the table by extracting through reflection the object properties and then populates the table with the values of the elements from the list.

**1.2. Application description**

The application allows a user to perform several operations on clients and products, like insert, delete update or see a the corresponding database in the form of a JTable. The user can also choose a client and a product and place an order for that client . A bill with the general information about this order will also be generated.

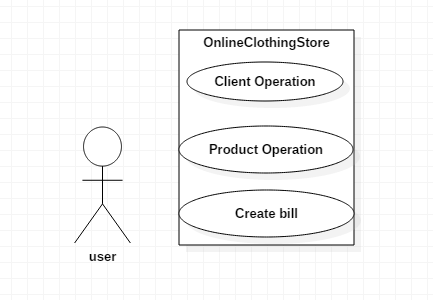
**2.Problem analysis and assumptions**

**2.1. General overview**

This application should be able to fulfill all the requirements in order to display, modify, and keep track of clients and products. The products ,clients and orders are stored in a relational MySQL database, along with the individual information and description of each attribute. This way, all the data is easier to retrieve and accessed from different computers.

**2.2. Assumptions**

In order for the application to work correctly and to avoid unhandled exceptions, some assumptions have to be made in respect to the input. Before attempting to start the simulation, the user should leave no input field unfilled. Moreover, we assume that each input is a positive, non-zero integer Also, the user should not attempt to input a product or client with an id that is already in the database and the user should also not input a string in the textfield of a integer (i.e a price,age or stock) .And the database tables should also contain at least one record for performing any kind of operations over it as this application doesn't use any kind of validators for the input data.

****

**3.Use cases:Simulate**

- Summary: This use case allows the user to simulate and observe the results of doing some simple querries over 2 tables of the database, Client and Products.

- Actors: The User

- Preconditions: None

- Main success scenario: 1. The user inputs the necessary data to start the simulation. 2. The user presses one of the buttons. 3. The application displays the table with the altered table , when Show table button is pressed. 4. The users observes the resulted records. 5. Simulation ends.

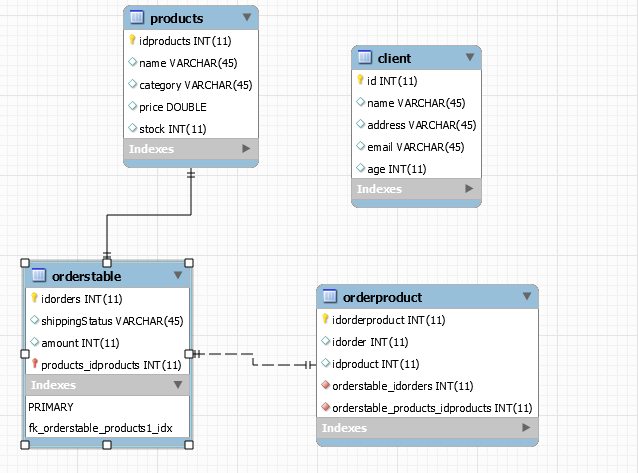
- Alternative sequences: None

- Error sequences: a) Invalid input: appears at step 2 1. The program returns an error; the use case fails.

**4.Design**

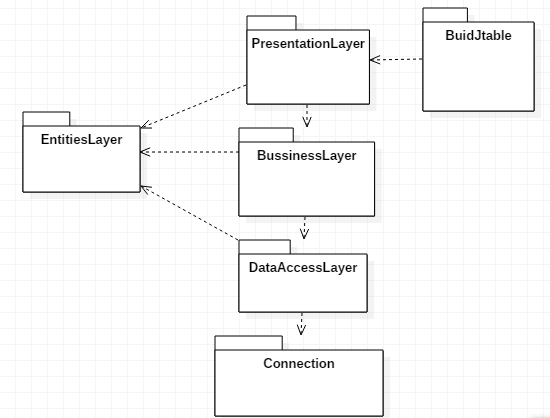
**4.1.Database**

At the base of the application stays a database containing the data on which the application operates. The design is very simple, consisting of three tables for the three main categories to be managed by the application.

****

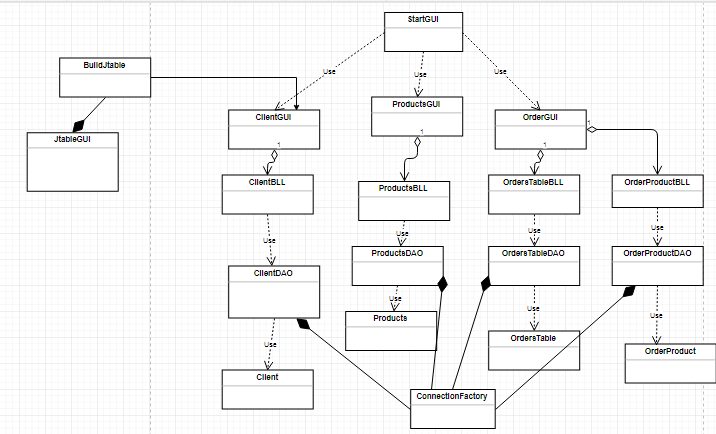
**4.2 Packages**

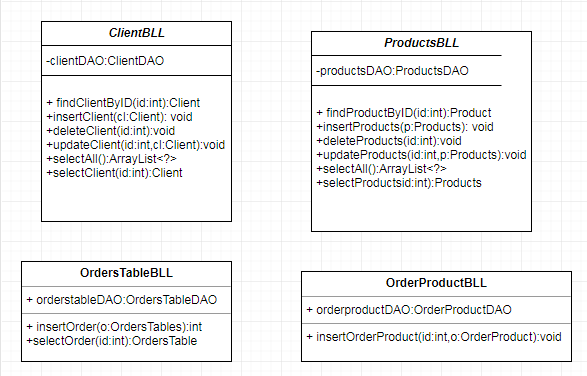
Packages represent these layers which are, respectively: the model layer, which contains the classes mapped onto the database, the data access layer (composed of packages connection and dao), which contains the classes which work on the database connection and construct the queries, the business layer (package bll), which contains the classes encapsulating the application logic and presentation layer which contains the classes defining the user interface.Another package named BuildJtable is used in order to create a generic JTable.A package which implements the logic that connects this application to the database is also present.

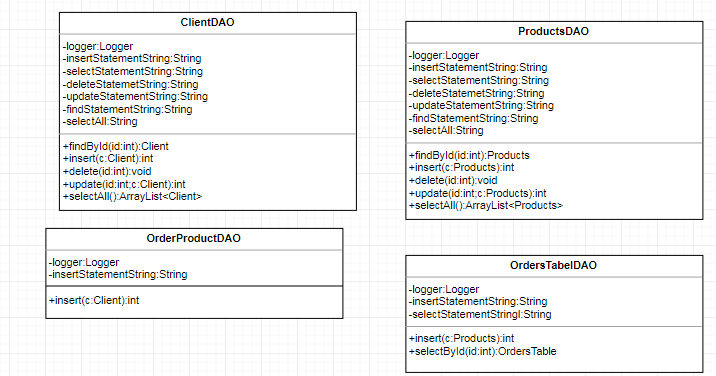
****

**4.3 Class design**

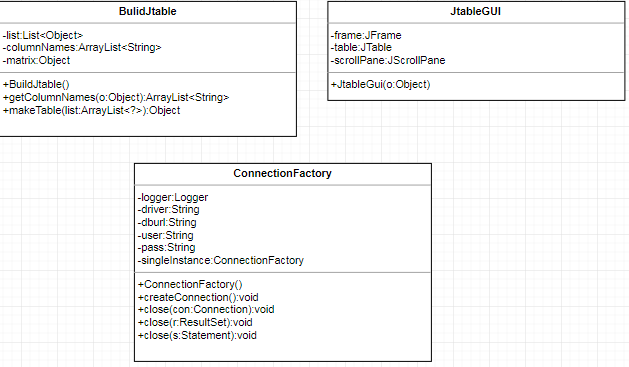
Class diagram is provided and the each layer will be analized in the following. As the design principles in certain layers are the same for the classes in that layer, class design will be detailed for each layer, not for each class, details of class design will be provided where necessary











**4.3.1 EntitiesLayer**

The classes from this package , Client, Order,OrdersTable and OrderProduct are used in order to provide acces to the data fields of the database and all 4 classes display an identical structure with the one of the corresponding table in the database.

**4.3.2 BussinessLayer**

Classes ClientBLL, OrderProductBLL,ProductsBLL and OrdersTable are using data from the EntitiesLayer and the DataAccesLayer in order to perform the insert, delete, update and select all operations.

**4.3.3 DataAccessLayer**

The classes ClientDAO, ProductsDAO,OrdersTableDAO and OrderProductDAO provide the methods for insert, delete , update and select all querries. . In order to ensure that these methods will also execute in the database, it uses the ConnectionFactory class to estabilish and close connections.

**4.3.4 Connection**

The most important class in this category is the ConnectionFactory one. It creates and closes connections with the database. The connection to the database will be placed in a singleton object, so only one instance of it can exist at a time. This is important, because if two queries are executed at the same time, one could affect the results of the other (ex. deletion and selection, the deleted element still appears in the selection).

**3.3.5. PresentationLayer classes**

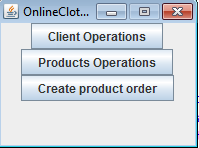
The classes defining the user interface are ClientGUI,ProductGUI,OrderGUI and StartGUI. They provide the interaction between the user and the program, showing data at request, and modifying it according to user input. Without these classes the average user wouldn’t be able to take advantage of the application.The StartGUI class represents the entry point of this application, from which the user will start the application.

**3.3.6.BuildJtable classes**

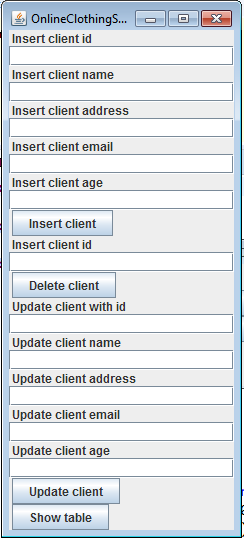
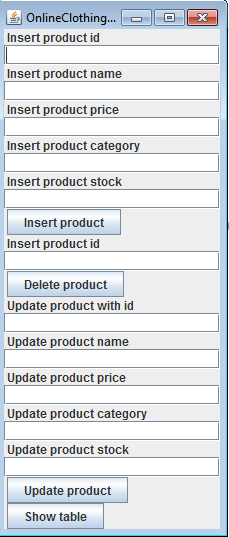
**The classes from this package , BuildJtable and JtableGUI provide the logic for creating an object of type JTable that will be used by the classes ProductsGUI and ClientGUI.**

**4.4 Graphical User Interface**

The application's GUI consists from 3 frames , which will provide the user the operation he/she can perform over the records of the database.The first window which appears describes the choices the user has , 3 options being presented here.

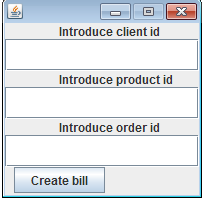
****

**The windows for the client and product operations are very similar, the only difference being the functionality and resultes obtained from it.**

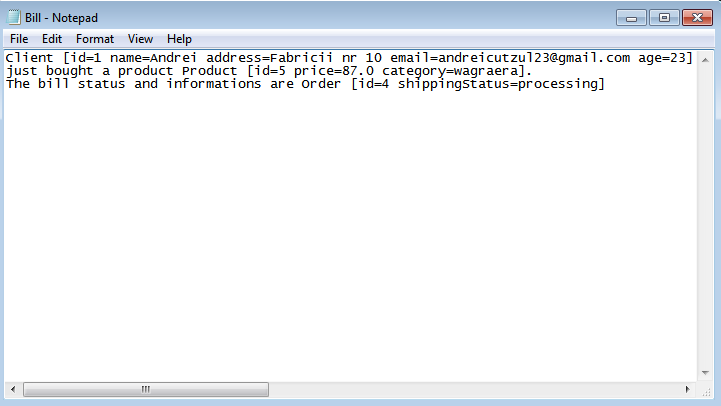
****

The Show table button is responsible with fetching the record from the database and dispalying them in the form of a tabel . The user can see in this way , after performing an operation, the changes over the database without having to enter MYSQL application .

If the user wants to place an order for a client , pressing the Create product order button will redirect it to a new frame which will allow him/her to select a client id , a product id and an order id. and therefore , to place a new order.

****

In order to see the bill created , the user should check the src folder from the package explorer , where a Bill.txt file is created and placed. If the amount of the products a client wants to buy is bigger that the number of available pieces of the selected product, the .txt file will contain a warning message that there are not enough products in stock.

****

5.Implementation

**5.1. The GUI**

The GUI consists of four windows, each of them providing means for user interaction. JButtons are used in the first window to open the other windows at user choice. In the ClientGUI and Product sGUI windows input is gathered from JTextFields, and there is no formatting on the JTextFields so input checking will be done at the press of a JButton. In these windows JButtons represent the action desired to perform on the input data, on success data will be cleared from the JTextFields and on fail warning messages will be displayed. In the ClientGUI and ProductGUI windows a JScrollPane containing a JTable is used, to display the entries of the database.

**5.2. Class implementations**

**5.2.1. EntititesLayer classes**

Each of the model classes represents an entry from a table in the database, so the data fields of a model class coincide with the columns of the data table. This way the Client class has the fields: id, name, address, email,age. The Products class: idproducts, name, category, price, stock and the OrdersTable class: idorders,shippingStatus, amount.The class OrderProduct is used in order to generate a 1 to many relationship and has idorderproduct,idproduct and idproduct. Each model class provides getter and setter methods for all of their fields, as well as a toString() method.

**5.2.2. BussinessLayer classes**

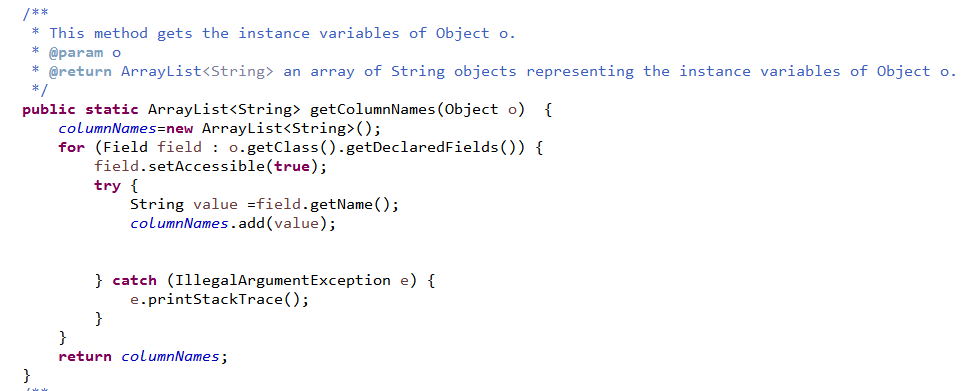
Each of the bussiness layer classes are used for implementing the logic prodived by the DataAccess layer classes

**5.2.3.DataAccessLayer classes**

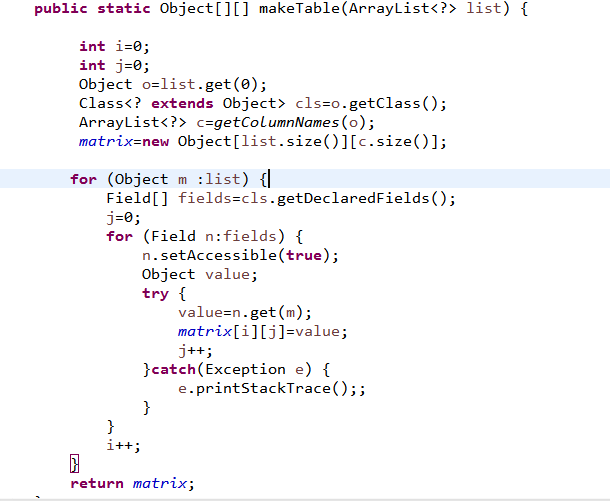
These classes are the core of the application, because here the querries are build and executed. Classes ClientDAO and ProductDAO provide methods for deleteting , inserting, updating and selecting all the products. Class OrdersTableDAO provides methods for select by id and insert in the database.Class OrderProductDAO provides a single method for insert in the table a new record, this class being used to keep track of all the orders placed by the clients.

**5.2.4.BuildJTable classes**

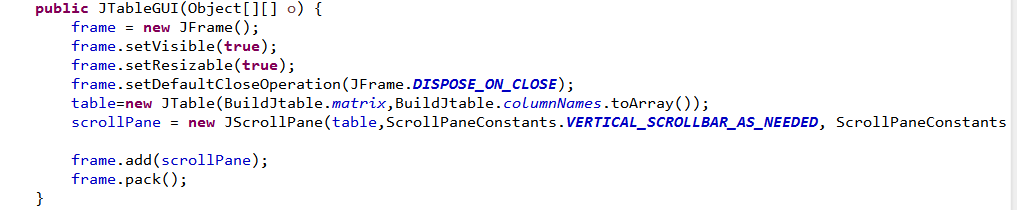
In the class BuildJtable from the buildtable package the main focus is on the method which returns the column names of an object o .



And the method which builds a matrix of the records from the database is presented below.



In the JtableGUI, with the object matrix created in the BuildJtable class a Jtable object is created, which will be placed on a frame with a scroll panel .



**5.2.5.Connection class**

**This class is provided for this assignment and I have not altered it's content , I used it in the same form.**

**5.2.6.PresentationLayer classes**

**This package is divided into 4 classes, StartGUI , which is the main class from which the application can be run, OrderGUI for placing an order and creating a bill, ProductGUI and ClientGUI for performing the main operations on the database.The classes ProductGUI and CLientGUI use JTextFields for taking the data which the user inputs and transforming it into the querries implemented in the DAO classes. This classes have access only on the BussinessLayer classes. The OrderGUI will display the 3 Jtextfiels in which the user is requested to input an id for selecting a client , a product and an order.The StartGUI class will call all the other GUI classes whenever the buttons are selecting an operation are pushed.**

6. Conclusions

**6.1. Further developments**

A useful development would be to add delivery companies to the database too, and make order deliveries possible. The application could be secured so that only the administrators have access to Client and Product s, but the regular user would be able to place orders for him/herself.Also , creating a profile secured with a password for each user can be further implemented in order to keep safe the informations of each user that perform operations in this database application.

**6.2. What I learned**

Working with this type of structure, layered architecture , was a tricky to understand at the begginning, but it seems a great idea that I will probably reuse in the future work and projects because it’s great for reusability and even for readability. Also I reckon that the building of the JTable was a problem that took me the greatest to solve from all assignment's request because I found it difficult to find the right solution that worked with my design. In conclusion , I enjoyed working for this assignment for it show me how to connect a Java program with a MySQL database and how to use Reflection when building a JTable.

BIBLIOGRAPHY

1. https://www.mkyong.com/java/how-to-write-to-file-in-java-bufferedwriter-example/ --> for creating the .txt bill

2.https://docs.oracle.com/javase/tutorial/uiswing/components/table.html#simple -->for getting familiarized with the structure and class of a JTable

3. https://docs.oracle.com/javase/7/docs/api/javax/swing/JScrollPane.html -->for getting used with the functionality of a ScrollPanel

4. https://kodejava.org/how-do-i-get-fields-of-a-class-object/ -->for building the array of String object for the instance fields of an Object