Assignment 3 Documentation

-Order Management-

**1.Homework objective:**

Main objective:

The task is to design an application for processing customer orders for a warehouse. Relational databases are used to store the products, the clients and the orders. Furthermore, the application should use (minimally) the following classes:

* Model classes – the data models for application
* Business Logic classes – implement the application logic
* Presentation classes – implement the user input/output
* Data access classes – implement the access to the database

Secondary tasks:

* The code should be implemented in the Object Oriented Programming design
* We need to use Javadoc for documenting the classes and generate the corresponding JavaDoc files
* The application should have a file parser for the commands.txt file
* The reports and the bills should be generated as PDF files
* The application should permit to be run with the following command:

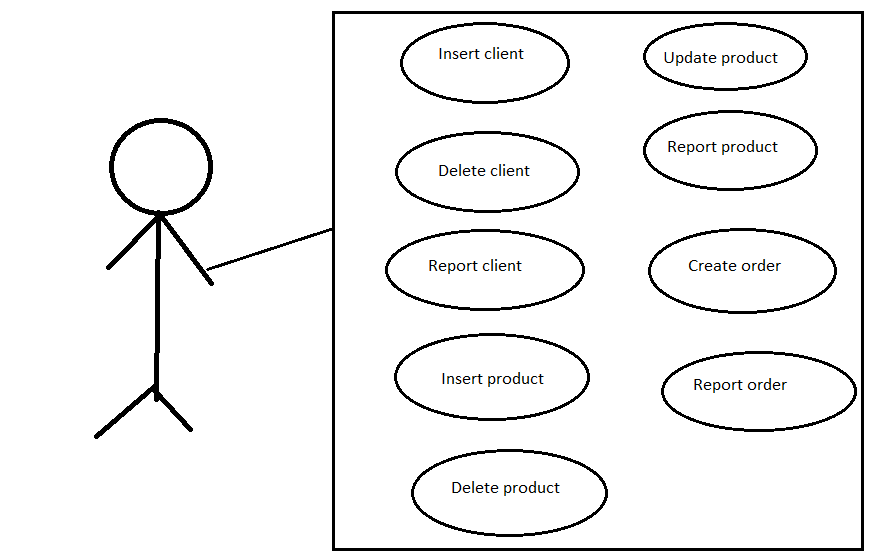
Java -jar pt2020\_30421\_roxana\_ciocian\_assignment\_3 commands.txt

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

Problem analysis:

We need to implement a correct and efficient program that takes as input a text file in which there are some commands that should be executed. The application should be able to fulfil all the requirements in order to display or modify orders, products and clients. The data is stored in a relational MySQL database. Talking about the input in this application, a user can choose to operate in 3 tables: Client, Product and Order. There are specified commands that can be executed in this application, such as: insert, delete, update or show all details, and the user can decide between those commands. The output will be generated as a PDF file when the command report will be executed, and it will display the content of the specified table.

Use cases:



Modeling:

This application needs as much models as we have tables in the database. So, we have three classes in the model package, each of them corresponding to a certain table from the database: Client, Product or Order.

Scenarios:

1. First scenario:
2. Identification summary:

Title: The command that should run the .jar file is wrongly introduced by the user.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the command

- the command is written with the wrong arguments that not correspond to the test that will be evaluated

- the program does not run as expected

- an error message appears in the terminal

2. Second scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Insert Client.

1. Flow of events:

-the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user, having a name for the new client and the city corresponding to his location

- the program runs as it is expected and the client is successfully introduced

3. Third scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Delete Client.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user, having a name for the new client and the city corresponding to his location

- the program runs as it is expected and the client is successfully deleted

4. 4th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Report Client.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user

- the program runs as it is expected and a PDF file containing the data from the Client table is successfully generated

5. 5th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Insert Product.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user, having a name for the new product, the quantity which will be on stock and the price for the product

- the program runs as it is expected and the product is successfully inserted in the table

generated

6. 6th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Delete Product.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user, having a name of the product to be deleted from the database

- the program runs as it is expected and the product is successfully deleted from the table

7. 7th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Report Product.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user

- the program runs as it is expected and a PDF file containing details about the Product table will be generated

8. 8th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Insert Order.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user, having a name of the client, the product he wants to buy and the quantity he needs

- the program runs as it is expected and the order is successfully inserted in the table

9. 9th scenario:

a.) Identification summary:

Title: The command that should run the .jar file is correctly introduced and the program executes the command from the text file, which is Report Order.

1. Flow of events:

- the user starts the program

- the user needs to rebuild the artifact before opening the terminal

- the user opens the terminal in order to introduce the correct command having the needed arguments

- the program will execute the command written by the user

- the program runs as it is expected and a PDF file containing the details about the Order table will be generated

**3.Desing and implementation:**

Data structures:

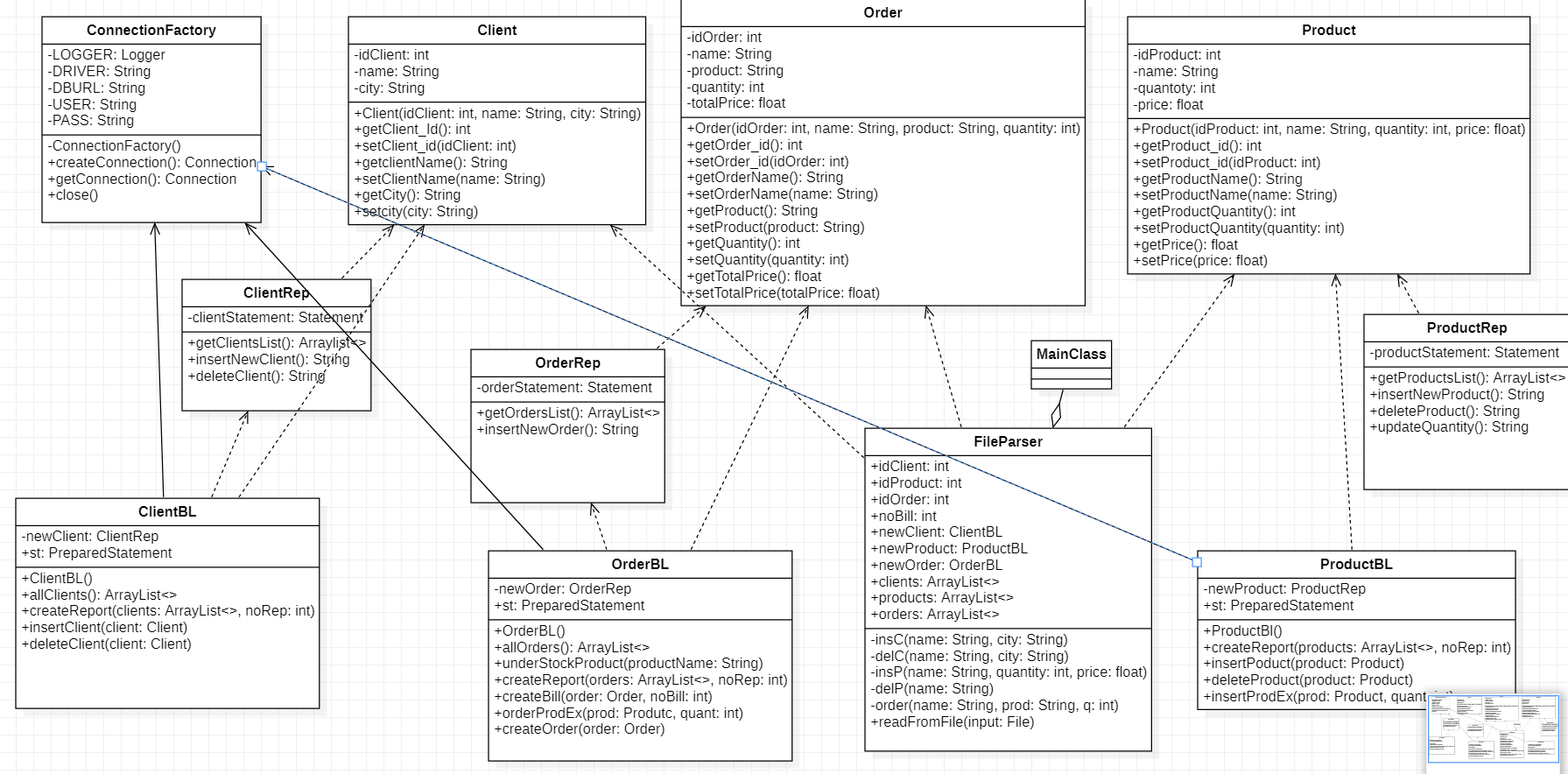
The only data structures used for this assignment is the list, and for the implementation of the list interface is used ArrayList. I chose to use this data, because this kind of lists are useful for storing and accessing data. This way I can keep track of all data from each table.

Packages:

We are asked to design the application such that we should have four main packages: Model, Presentation, DataAccess and BusinessLogic.

* Presentation layer: in this package should be put the classes that implements the input/output
* Model layer: in this package should be put the classes that are mapped to the database table
* DataAccess layer: in this package should be put the classes containing the queries and the connection to the database
* BusinessLogic layer: in this package should be put the classes that encapsulate the application logic

Uml diagram:



Class design:

This application is based on the main three classes: Client, Product and Order, which represents the mapped classes to the database

The design has the format:

* Model:
* Client - this class models the client table from the database
* Order – this class models the order table from the database
* Product – this class models the product table from the database
* DataAcces:
* ConnectionFactory – this class is responsible with the connection to the database. There are also methods for closing the connection, statement, prepared statement and result set
* ClientRep – this class is responsible for creating the queries that corresponds to the client table
* ProductRep – this class is responsible for creating the queries that corresponds to the product table
* OrderRep – this class is responsible for creating the queries that corresponds to the order table
* BusinessLogic:
* ClientBL – this class is responsible for the operations that takes place for the Client table. Here are created the methods for inserting and deleting a client, a method for creating an array list of clients which contains the data from the table, and also, here is the method for generating a report for the Client table
* ProductBL – this class is responsible for the operations that takes place for the Product table. Here are created the methods for inserting, deleting and updating a client, a method for creating an array list of products which contains the data from the table, an also the methods for generating a report for the Product table
* OrderBL – this class is responsible for the operation that takes place for the Order table. Here are created the following methods: a method for creating an order, a method for updating the product quantity after the order is created, a method for creating an array list of orders which contains the data from the table, and also methods for generating a report for the table, for generating an order bill and for generating an under stock PDF file
* Presentation:
* FileParser – this class is responsible for getting a file and read the lines from it. Also, here are executed the command from the file and the reports are generated.
* MainClass – this class is responsible for running the application for the certain commands.txt file given as an input

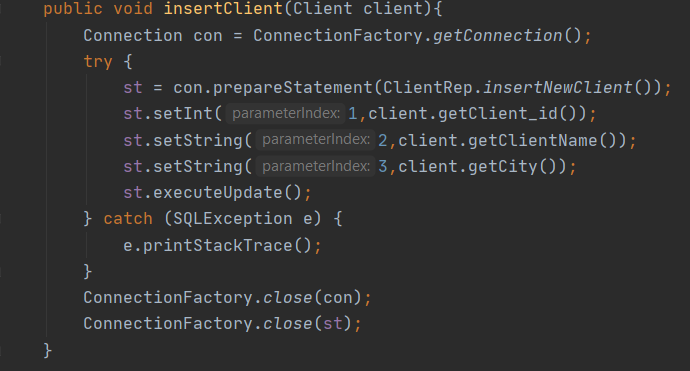
Implementation:

The classes for the Model package are characterized by the fields corresponding to each table, setters and getters, and also a toString function which returns the clients, products or orders into a normal form. Moreover, the Order class has an additional parameter for the total price of the order, having also getter and setter.

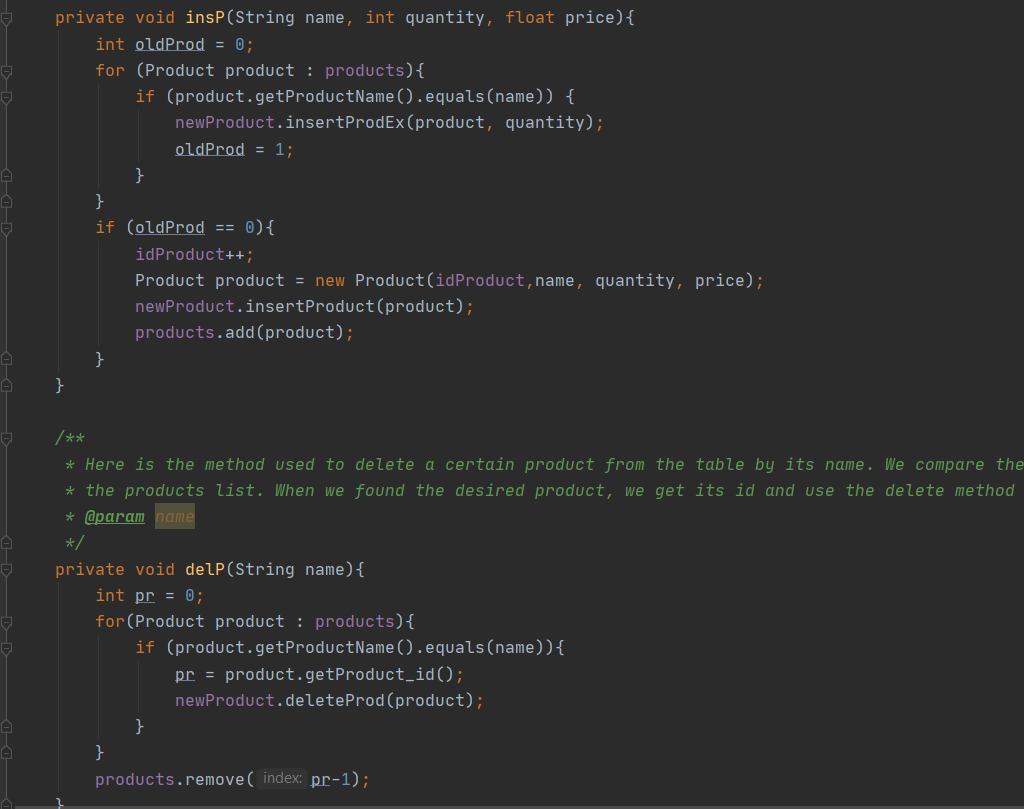
In the classes from the Data Access package, for example, in the ClientRep class, there is a method for extracting each element from the table and creates an array list o such elements. The rest of the methods represents the queries for some command instructions of a certain table. Similarly, these methods are found in the rest of the classes.



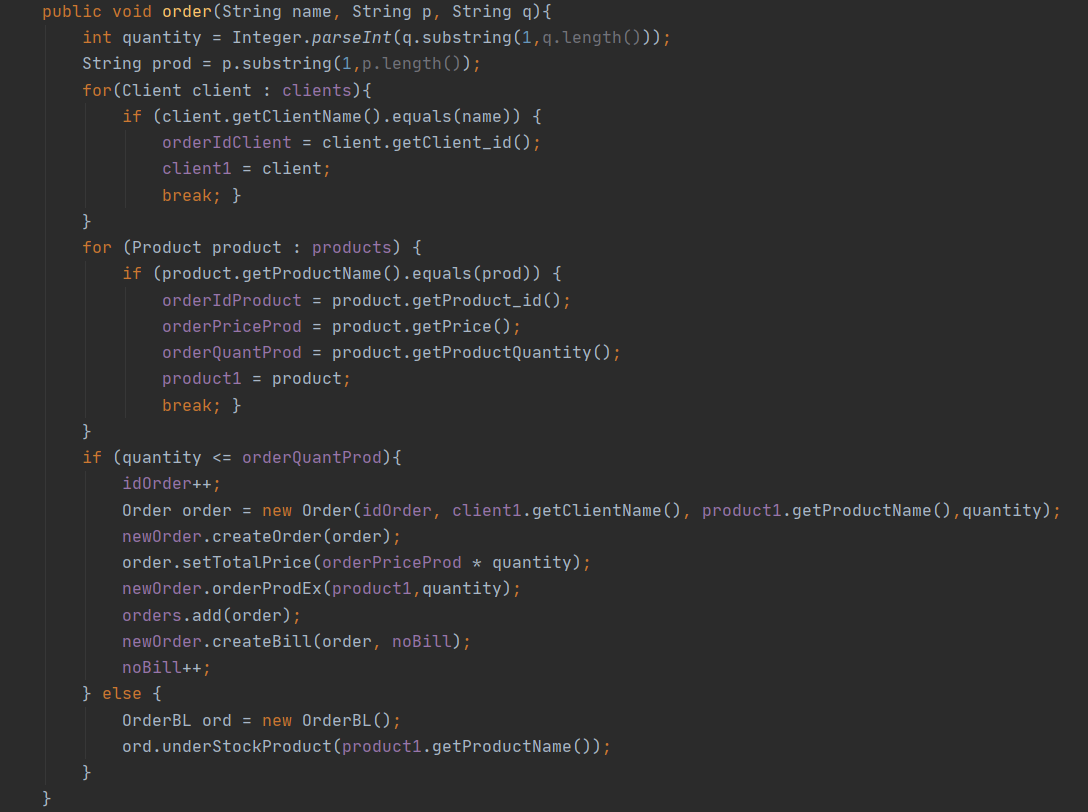
The classes from the Business Logic package implements the same logic, but on different circumstances. For example, in the ClientBL class, there are two methods, one for inserting a client, and one for deleting a client. Both of them have the same logic, but they implement different queries. This method sets the parameters of the client which will be inserted. Also, here are the methods for generating the reports. In the rest of the classes, there will be approximatively the same methods, but implementing different queries.



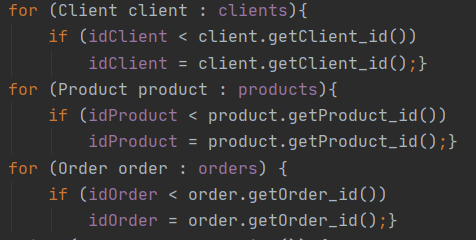
In the Presentation package, there is only one class, File Parses, where the input file is read and the commands are executed. This class contains methods for executing the commands. For example, the methods for inserting/deleting/updating a product. When we want to insert a product by its name, quantity and price, we have to check if that product already exists in the table. For doing that, we need to go through the products list, and search for a given name,. If there already exists that product, we just update its quantity, using the method that executes the update query. If the product does not exist, it will be created using the method which executes the insert query. For deleting a product by its name, we need to go through the list of product again and find that product, getting its id, and delete that product by its id using the method that executes the delete query.



When inserting an order by its name, product name and quantity, we need to find the client in the list o the clients and make a copy of it, same with the product, storing also its quantity and price. After that, we need to check if there are enough quantity of that product on the stock. If it is, we create the order by calling the method which executes the insert query, and also we create a bill of that order. If the quantity is not enough, the a under stock pdf of that product will be generated, and the order will not be inserted.



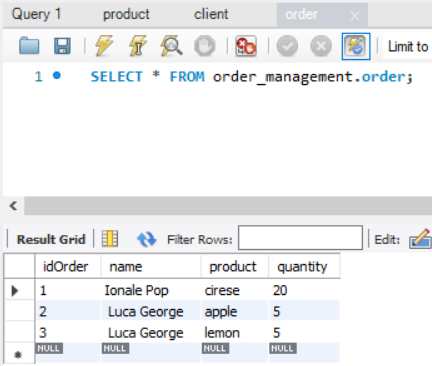
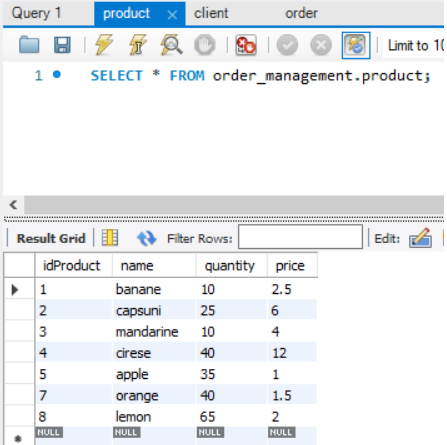
The id of the position where the elements can be inserted is found by the following code. We check the id of the elements from the list and take the greater one.

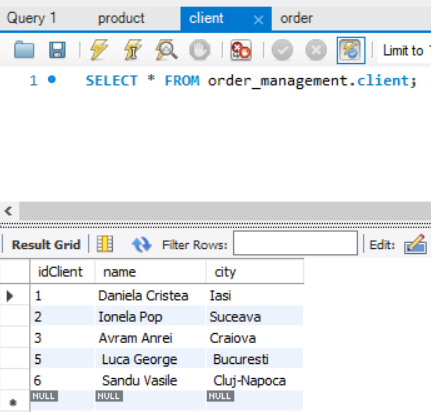


Also, the commands are read and executed in the following way. We split the line by “:”, so in that way we will have in the first string the command that should be executed and in the second one we will have the parameters.



**5.Results:**





**6.Colcusions:**

This project was a good opportunity for me to work with MySQL and try to connect a database to a java project, and also a good opportunity to learn how to generate different kind of outputs, such as PDF files. I managed to run the .jar in the IntelliJ terminal with the command specified in the task, which is java -jar pt2020\_30421\_roxana\_ciocian\_assignment\_3.jar commands.txt.

**7.Bibliography:**

<https://howtodoinjava.com/library/read-generate-pdf-java-itext/>

<https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>

<https://www.javatpoint.com/example-to-connect-to-the-mysql-database>