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:

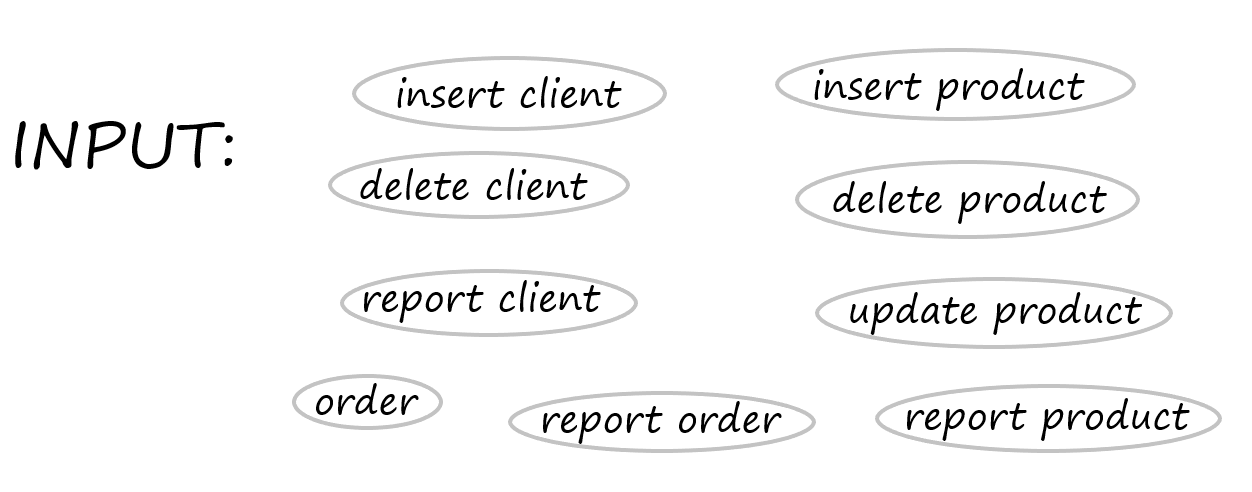
Java -jar pt2020\_30421\_Roland\_Soos\_assignment\_3 input.txt

**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 4 tables: Client, Stock, Order and Bill. 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 four classes in the model package, each of them corresponding to a certain table from the database: Client, Product, Order or Bill.

Scenarios:

1. First scenario:
2. Identification summary:

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

1. Flow of events:

- the user starts the program

- 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 program 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

- a success message appears in the terminal

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 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 tables

- this command actually inserts data into the Order table and the Bill table as well

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 and a PDF file containing details about the Bill table will be generated

**3.Desing and implementation:**

Data structures:

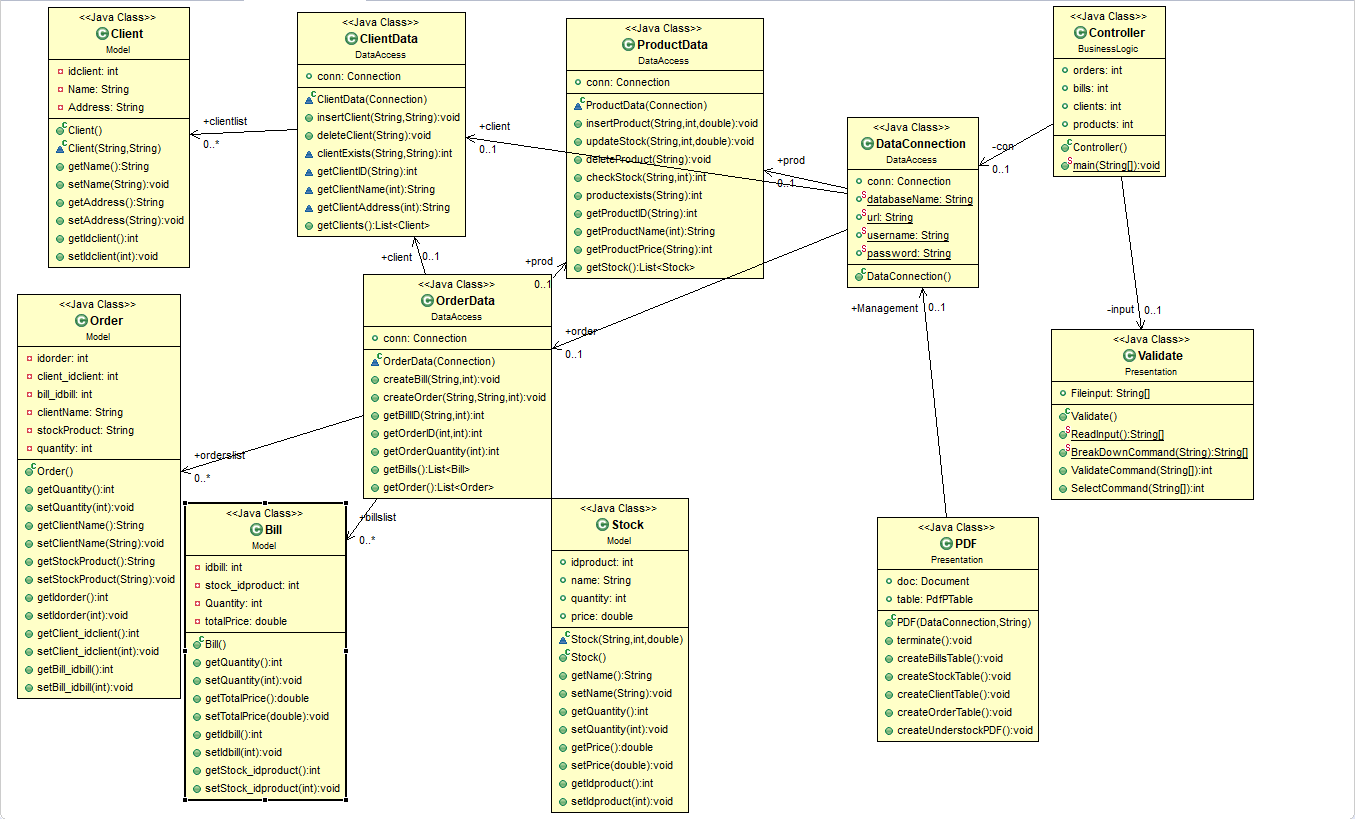
The data structures used in this project are List and ArrayList. These are very helpful because they can store objects like client or order, thus making it easy to keep track of the data. I used these lists of objects to create all my table from the PDF files.

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 four classes: Client, Stock, Order and Bill, which represents the mapped classes to the database

The design has the format:

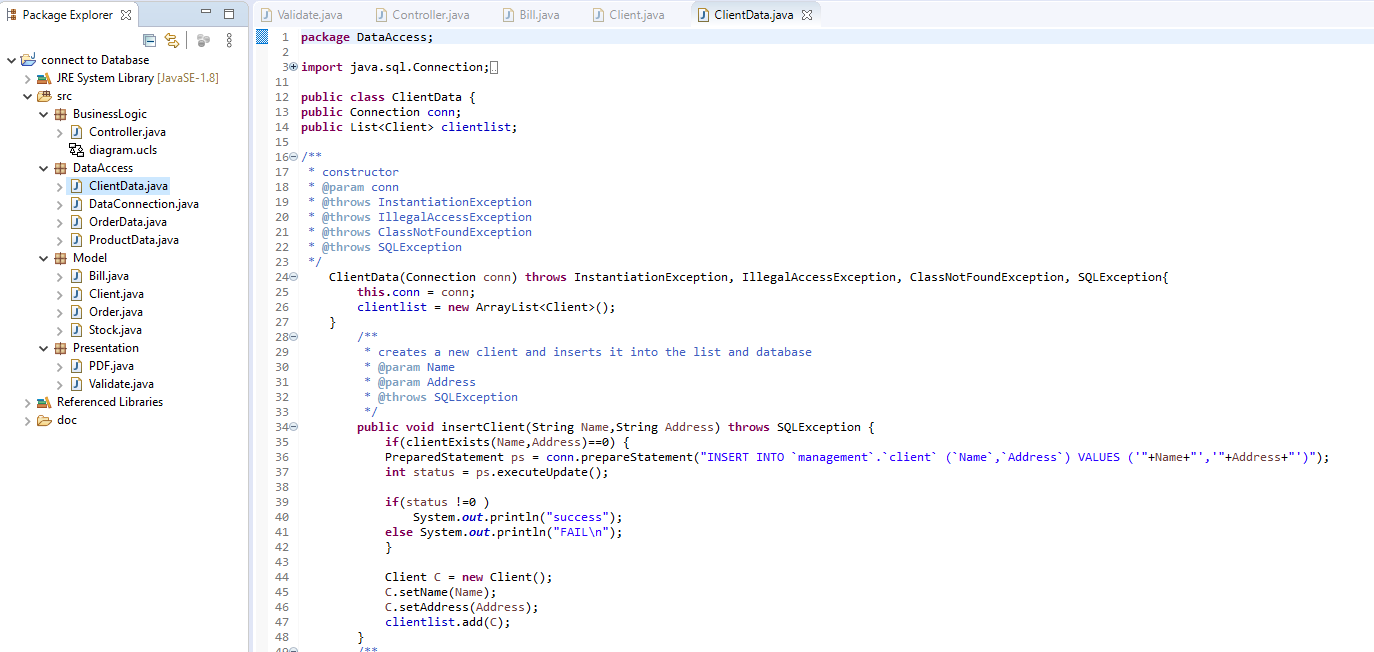
* Model:
* Client - this class models the client table from the database
* Order – this class models the order table from the database
* Stock – this class models the product table from the database
* Bill – this class models the bill table from the database
* DataAcces:
* DataConnection – this class is responsible with the connection to the database.
* ClientData – this class is responsible for creating the queries that corresponds to the client table and creating a list of all the clients
* ProductData – this class is responsible for creating the queries that corresponds to the stock table and creating a list of all the products
* OrderData – this class is responsible for creating the queries that corresponds to the order and bill table and creating lists with all the orders and bills
* BusinessLogic:
* Controller – this class is responsible for the logic of the whole project and running the application
* Presentation:
* Validate – this class is responsible for getting a file and read the lines from it. Also, here are checked the commands read from the file.
* PDF – this class is responsible for the output of the application, more specifically the PDF files. Here are the tables filled with the lists of objects.

Implementation:

The classes for the Model package are characterized by the fields corresponding to each table, setters and getters. Their role is to be used as objects that will be kept in lists and inserted in the tables of the PDF files.

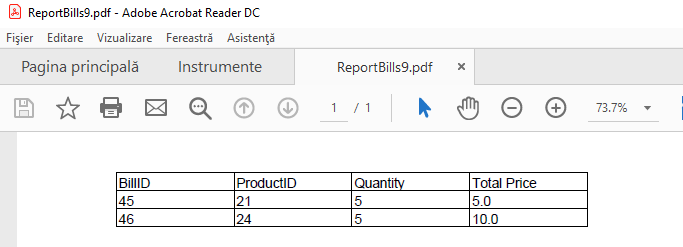


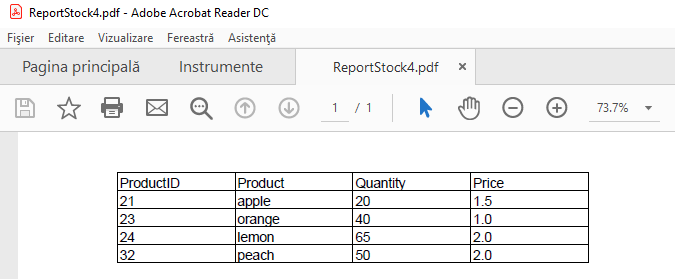
In the classes from the Data Access package have methods with MySQL queries that will either use the data from the database or insert new data into the database. They also create objects and store them in lists of objects. Doing both of these action make your life easier of keeping track of the data.

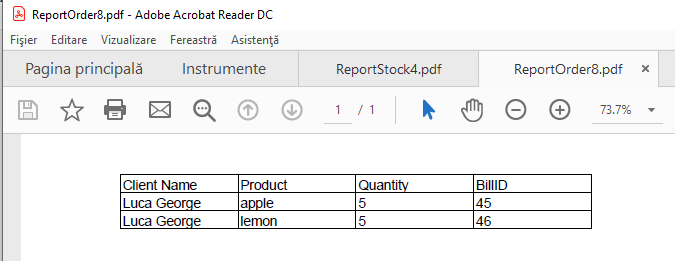


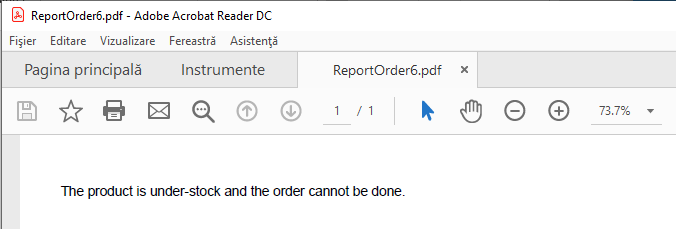


When inserting an order by its name, product name and quantity, we need to find the client in the list o the clients and make a copy of it, same with the product, storing also its quantity and price. After that, we need to check if there are enough quantity of that product on the stock. If it is, we 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.

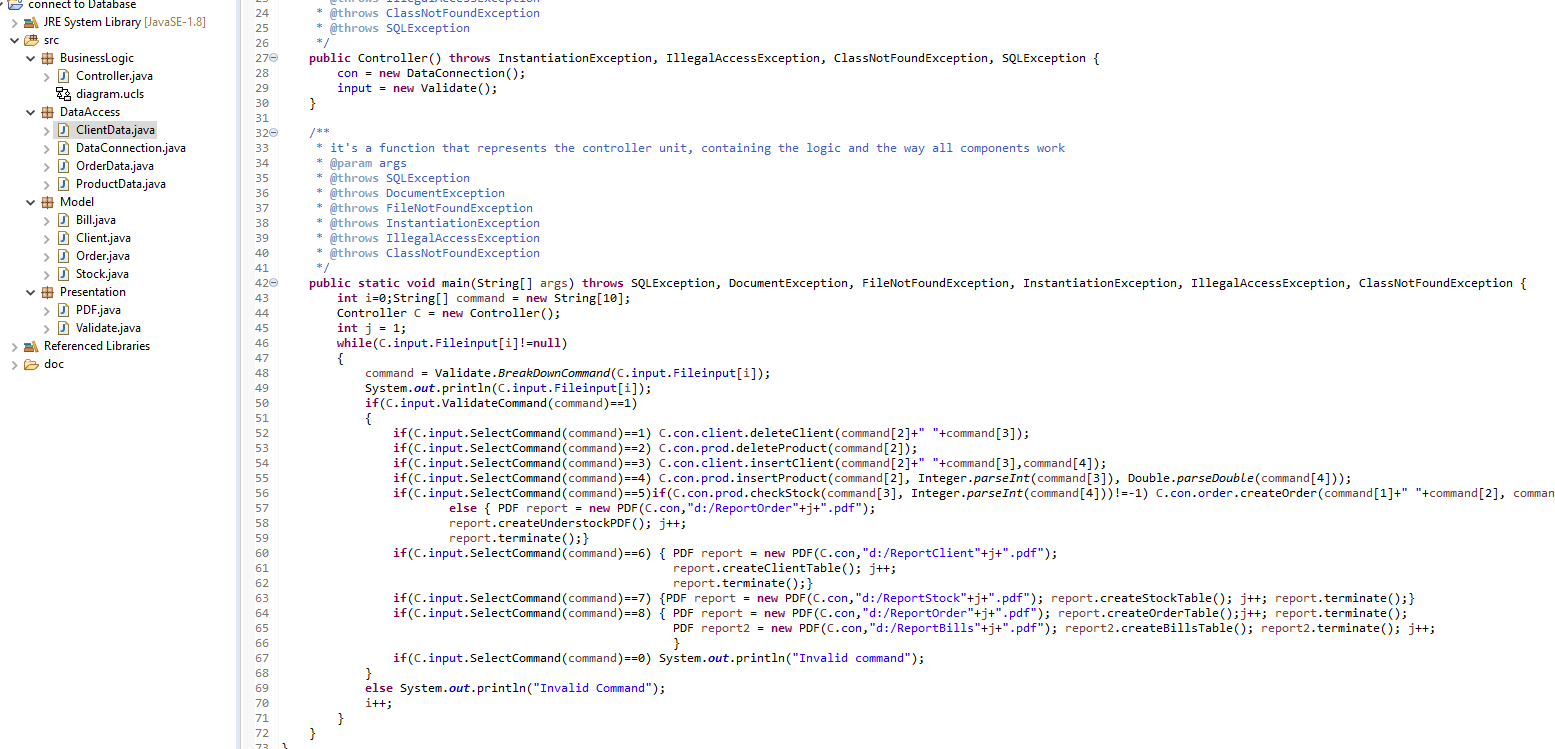




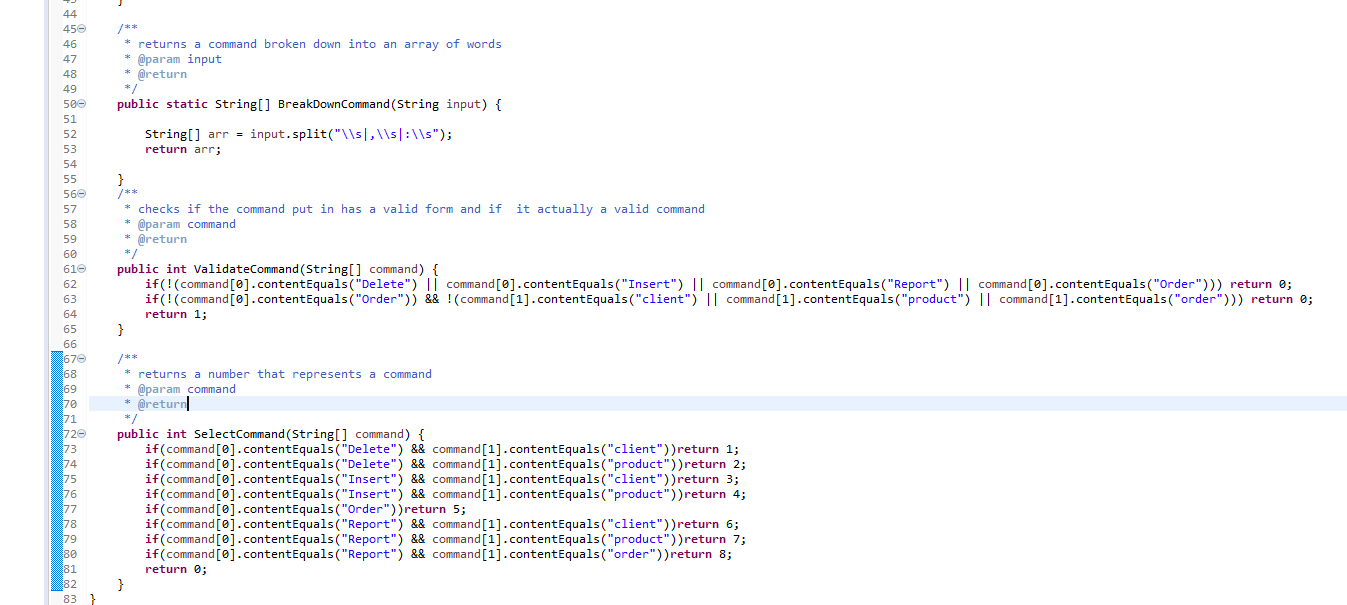




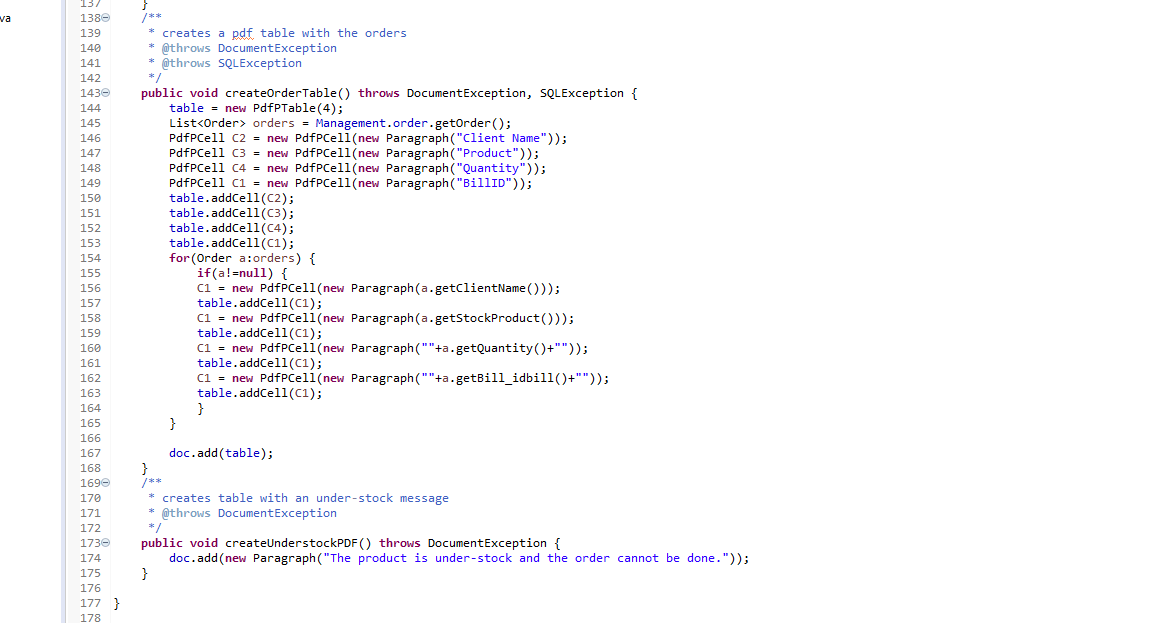
The BusinessLogic package contains only one class, called Controller, which is the main class that runs the application. Using the Validate class, it reads the input file, checks the commands, based on the command output as a number (for example 1 is for insert client) it decides what to do next. After each command will appear at least one message either “success” or “Invalid command”.



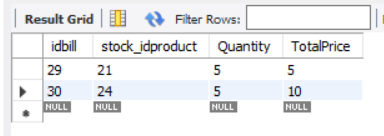
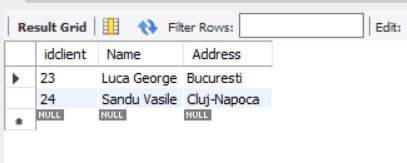
In the Presentation package, there are two classes: Validate and PDF. The Validate class has a method that parses the input file called “input.txt” and returns every line read as a String, then a method will break down each line and send it to the method that checks if the command has a valid form and if the command exists, finally finishing with a method that returns a number that corresponds to a certain command. The lines are split using a regex pattern "\\s|,\\s|:\\s"

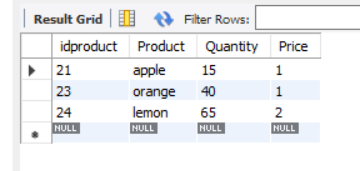
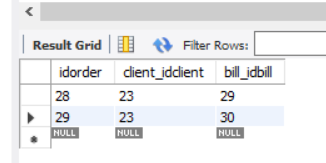


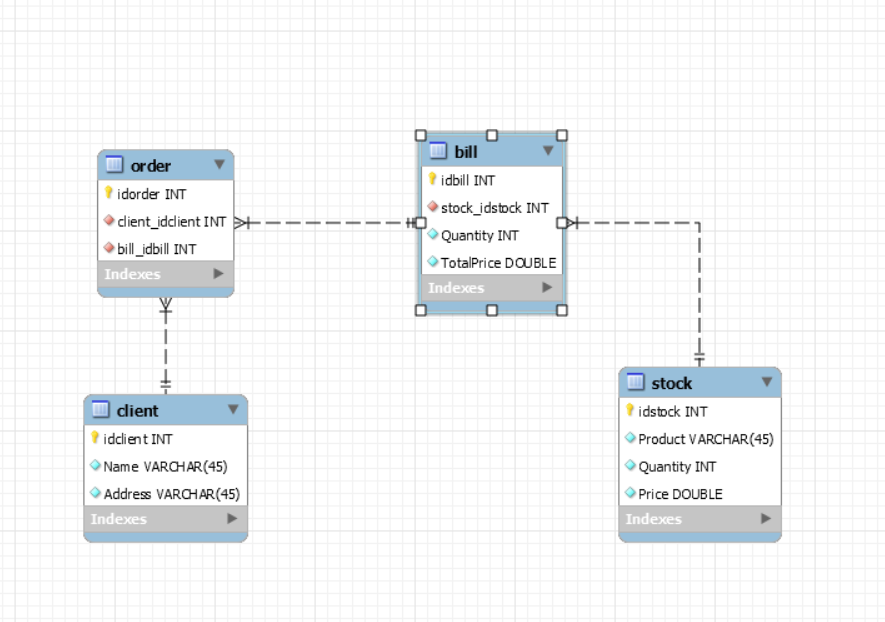
The PDF class has methods that will fill up the tables of the PDF file, open the PDF file and close it after the tables are put in. When attempting to fill a table, the data is received using a method from the DataAccess package, for example: getOrders is a method in the OrderData class from the DataAccess package and it returns a list of Order objects.



**5.Results:**

****

****

****

**6.Colcusions:**

This project was a great opportunity to learn how to connect MySql database with my Eclipse project and run everything in java code. It was a little challenging, but I think it was very helpful.

**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>