DOCUMENTATION

ASSIGNMENT *NO_3*

STUDENT NAME: Ioana-Emilia Terente
GROUP: 30422

CONTENTS

1. Assignment Objective

The main objective of the assignment is to design and implement an application for managing a client's orders for a warehouse.

The sub-objectives of the project are to:

- Analyze the problem and identify the requirements
- Design the orders management application
- Implement the orders management application
- Test the orders management application

2. Problem Analysis, Modeling, Scenarios, Use Cases

Problem Analysis:

This application should be able to fulfil all the requirements in order to display, modify, and keep track of orders, clients and products. These are stored in a relational Postgressql database, along with the information which the user can have access to in the system.

When talking about the input in the application, the user can choose to manage 3 tables, Client, Product and Order. All tables have the options defined as the CRUD operations, Create (insert entry), Read (show all entries), Update (modify entry), Delete and viewAll, and the user can introduce the values in the specific fields for any of the three different tables.

For example, for the Client table the user can introduce an ID for the client (if no id is provided the application will return an error), the name of the client, the phone number and the email address. When inserting data into the tables all fields must contain some data or the program will return an error that the field cannot be empty.
For the following tables, the input is similar but with different fields.

Primary Actor: employee

Use Case: product and client management with order placement

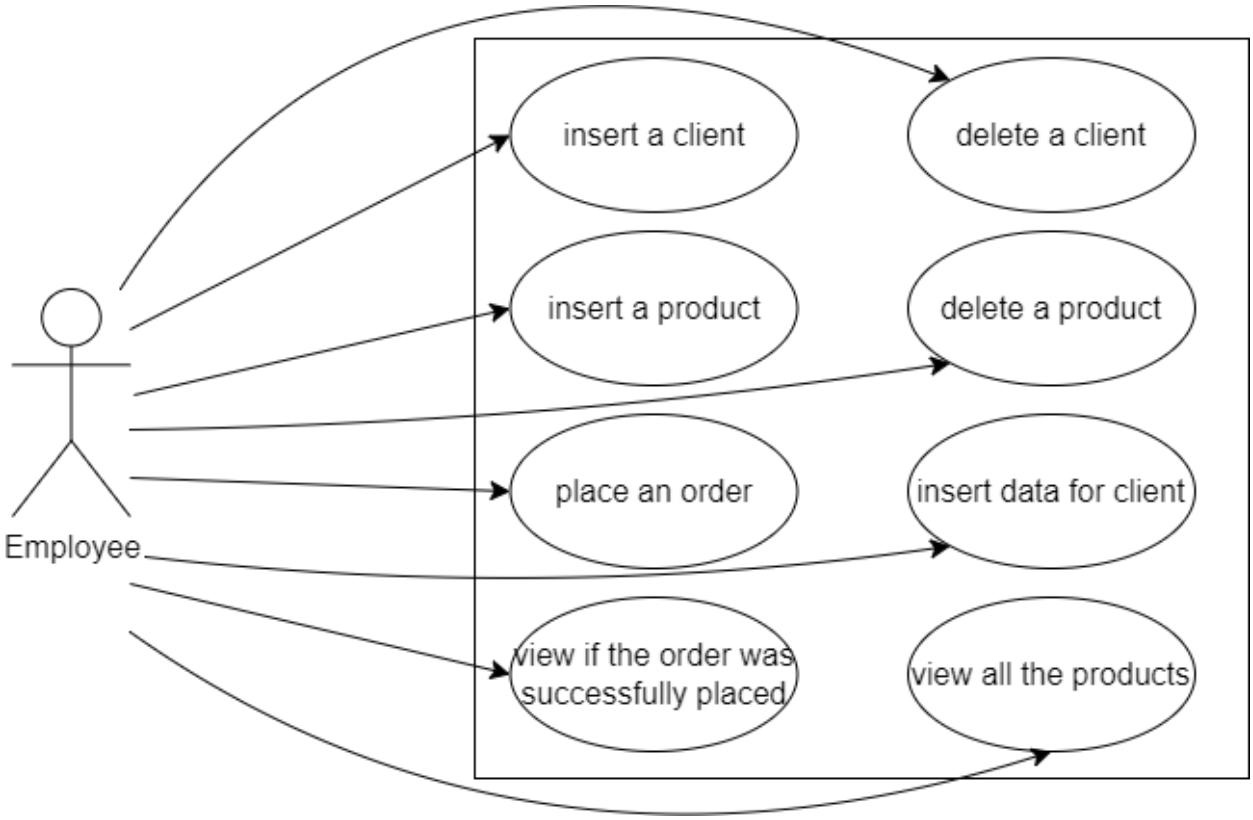
Modelling:

Main Success Scenario:

1. The employee selects the option to add a new product
2. The application will display 4 fields in which the product details will be inserted: id, name, price and stock
3. The employee inserts the id of the product, its name, price and current stock
4. The employee clicks on the "Insert" button
5. The application stores the product data in the database and displays the table with all products
6. The same steps are for the option to add a new client, windows can be open in the same time
7. The "Back" button is pressed and the initial page is accessed
8. The employee can set an order by inserting all the necessary data
9. If the order was successfully placed then an informational message will appear on the screen

Alternative Sequence:

- If not all the fields are filled, then the program will show an error that fields can be left empty
- If the employee introduces an existing id, then the program will display an error that the specific id already exits
- If the employee sets an order with a product that is out of stock then an informational message will be displayed on the screen



Functional Requirements:

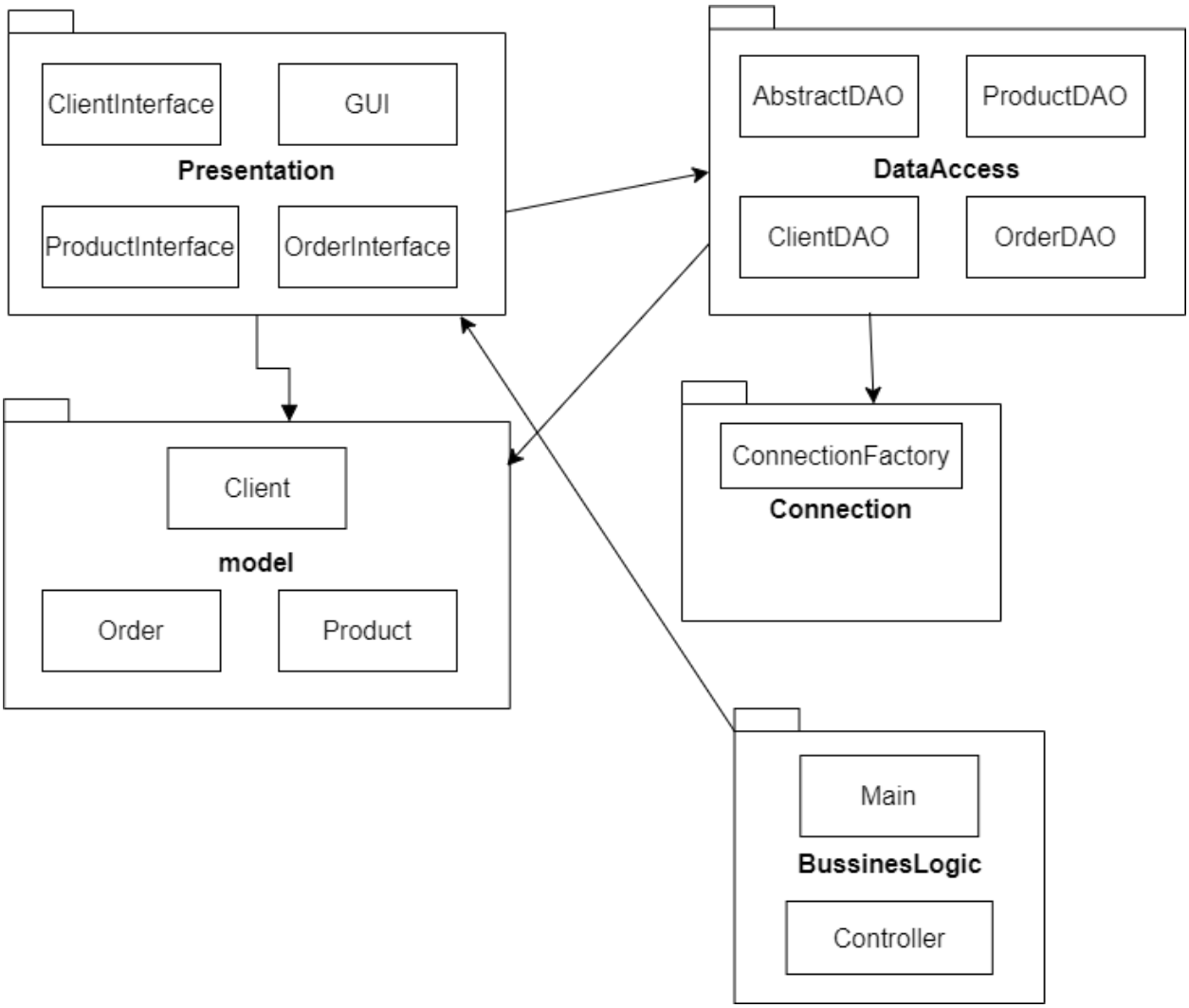
- The application should allow an employee to add a new client
- The application should allow an employee to add a new product
- The application should allow an employee to view all the clients from the database
- The application should allow an employee to view all the products from the database
- The application should allow an employee to delete a client from the database
- The application should allow an employee to delete a product from the database
- The application should allow an employee to set m order

Non-Functional Requirements:

- The application should be intuitive and easy to use by the user, the user can easily insert values in the text fields for each table: client product and order
- The application should be able to handle a large number of clients and products and distribute them properly
- Liability
- Scalability

3. Design

UML Package Diagram



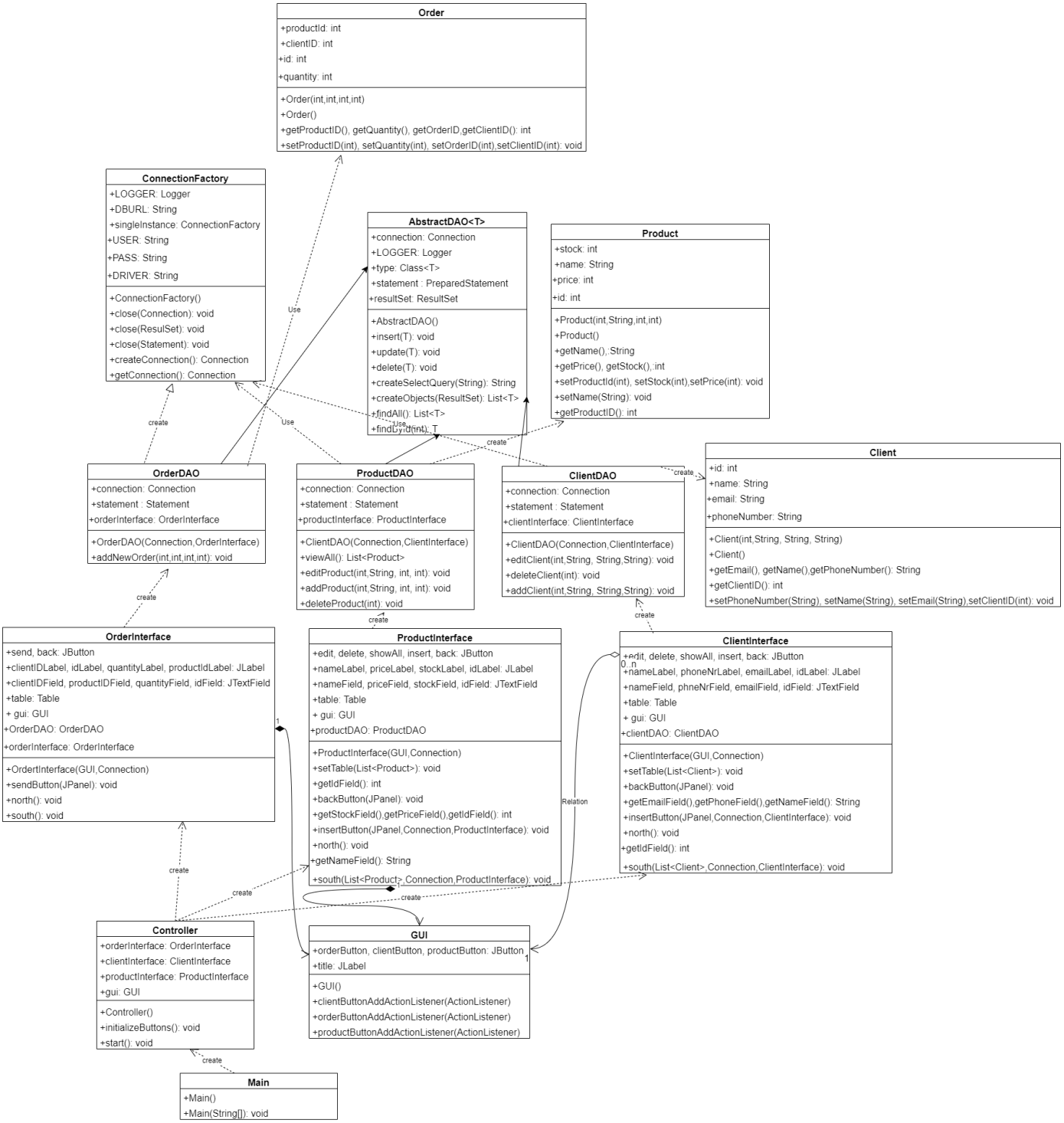
Packages:

Java packages help in organizing multiple modules and group together related classes and interfaces.
In object-oriented programming development, there are multiple packages:
- *model*: which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.

- *presentation*: which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)

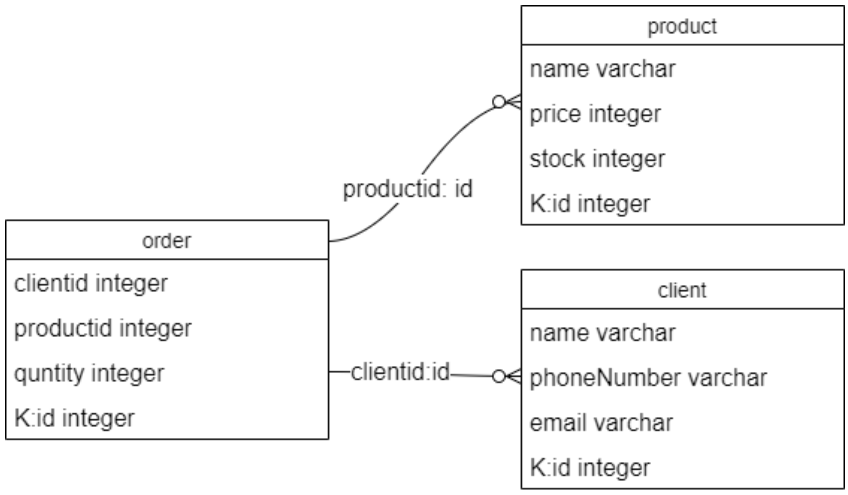
-*controller*: which represents the classes connecting the model and the view and is used to communicate between classes in the model and presentation.

UML Class Diagram



4. Implementation

Database diagram



Class Design

I have divided my program into multiple classes so that is easier to divide the problem into smaller problems and solve them.

BussinessLogic

- *Controller:* initialize the buttons from the GUI and makes the other interfaces visible when the correspondent button is pressed
- *Main :* calls the Controller class and runs the application

Connection:

- *ConnectionFactory:* creates the connection with the tables from database

DataAcces:

- *AbstractDAO:* has generic methods that implements the basic operations: insert, delete, update and viewAll by appending queries. All generic methods use reflection techniques.
- *ClientDAO:* extends the AbstractDAO and uses the generic methods to implement CRUD operations for the Client table.
- *ProductDAO:* extends the AbstractDAO and uses the generic methods to implement CRUD operations for the Product table.
- *OrderDAO:* extends the AbstractDAO and uses the generic methods to implement CRUD operations for the Order table.

Model:

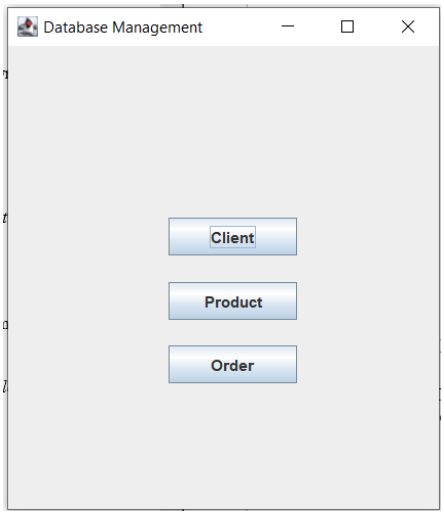
- *Client:* models the Client table from the database
- *Order:* models the Order table from the database
- *Product:* models the Product table from the database

Presentation:

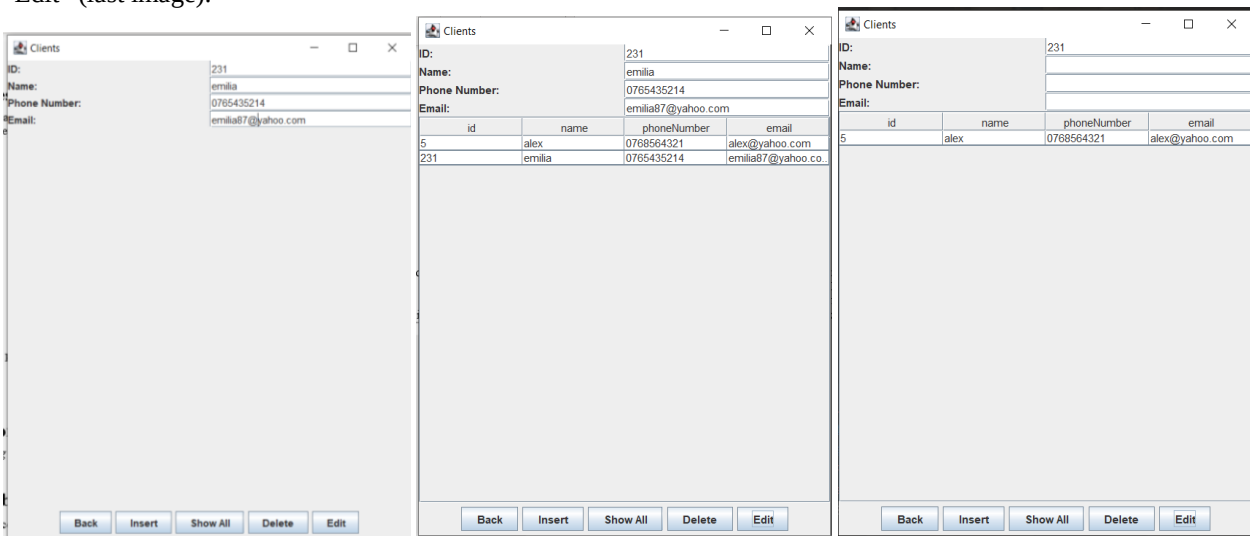
- *ClientInterface: C*lient frame, provides buttons, labels, text fields and creates a JTable using reflection techniques that has the data from the table
- *ProductInterface:* Product frame, provides buttons, labels, text fields and creates a JTable using reflection techniques that has the data from the table
- *OrderInterface:* Order Product frame, provides buttons, labels, text fields and creates a JTable using reflection techniques that has the data from the table
- *GUI:* the main frame with is the “parent” of all the other frames in this application. Provides access to the other frames.

5. Results

The graphical user interface is easy to use and understand by the employee, the buttons have written clearly on them what they are doing.

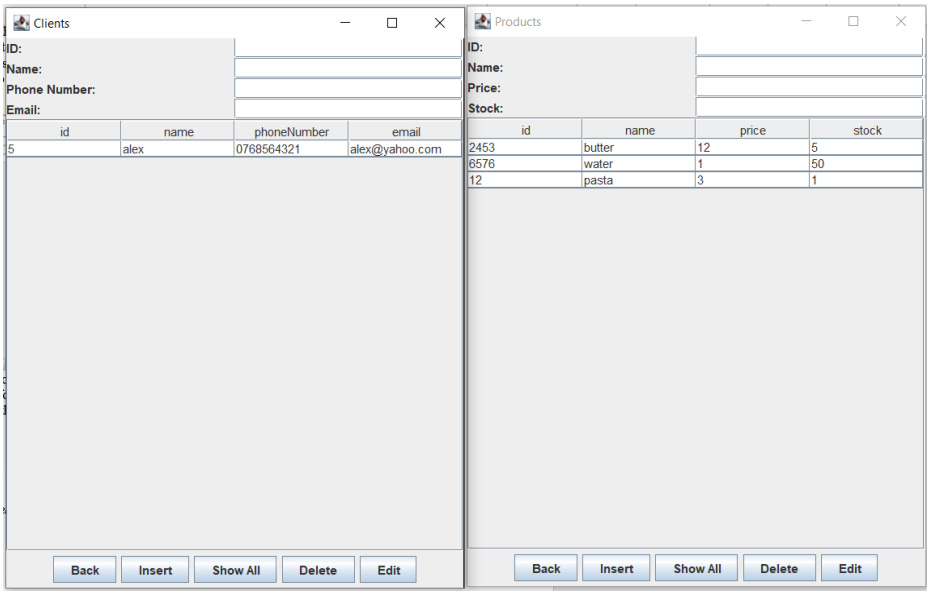


Each: Client, Product or Order page can pe opened in parallel such that the employee can see al the data from the tables.Here is the initial page with some data introduced by the employee before pressing any button. To add the client to the table it is necessary to press the “Insert” button. The graphical user interface will remain the same but the client will be introduced. To see the table with clients press “Edit” button and the page will look like this (middle picture).If the employee wants to delete any client it shoul insert its id and press the “Delete” button followed by the “Edit” (last image).

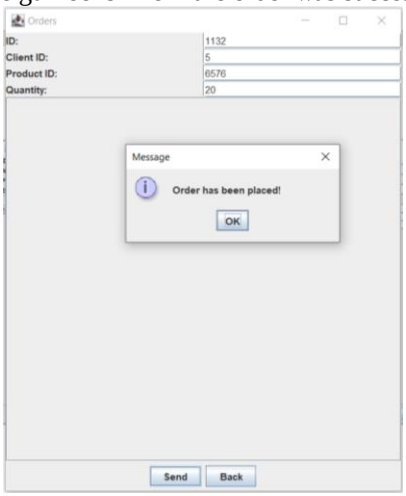


The employee should take into consideration that a client that has placed an order can not be deleted from the table. The above example is the same for the Product graphical interface.

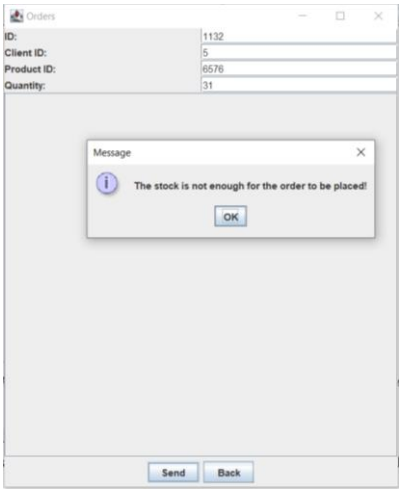
Here is an example for an order that has been placed, here are the products and the clients displayed by pressing the “Show All” button.



This is how the gui looks like if the order was successfully placed:



And this is how it looks like if the warehouse run out of stock:



6. Conclusions

This project was a good exercise in remembering the OOP concepts learned in the first semester, but also learning new ones, I found it very useful and challenging at first. There are a few learned things which I would present next.

First of all , time management crucial, because a good organization helps you see things gradually and making things from time helps you a lot.

Secondly, modelling the problem in a right way from the beginning helps you to implement it faster.

Thirdly, I arrived at the conclusion that facing problems with your code and trying to make it work by yourself, through the mean of research, has the benefit of learning new concepts and a better use of the known ones.

By the means of this project I managed to improve my knowledge about reflection techniques and how they are implemented on a system, I learned about database connection and how to improve my overall knowledge abou the OOP concepts.

7. Bibliography

1. Fundamental Programming Techniques: https://dsrl.eu/courses/pt/
2. What is JavaDoc tool and how to use it? https://www.geeksforgeeks.org/what-is-javadoc-tool-and-how-to-use-it/
3. DAO Class in Java https://www.javatpoint.com/dao-class-in-java
4. CRUD Operations in Java https://www.geeksforgeeks.org/crud-operations-in-student-management-system-in-java/