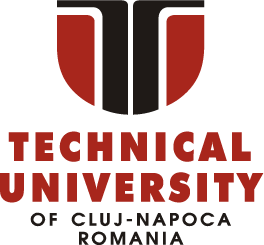
Technical University of Cluj-Napoca

Programming Techniques

Laboratory Assignment Three Documentation

Order Management

**

Teacher: prof. Dr. Ioan Salomie

Laboratory teacher: Dr. Cristina Bianca Pop

Student: Nemeș Mihnea-Andrei

Group: E\_30424

Contents

[1.Assignment Objective 3](#_Toc70394452)

[2.Problem Analysis, modelling, scenarios and use cases 3](#_Toc70394453)

[3.Design 6](#_Toc70394454)

[4.Implementation 9](#_Toc70394455)

[5.Results 10](#_Toc70394456)

[6.Conclusions 11](#_Toc70394457)

[7.Bibliography 11](#_Toc70394458)

# 1.Assignment Objective

For this assignment we have to consider an application that focuses on Order Management, for processing client orders for a warehouse. Relational databases are to be used to store the information about the clients, the products, and the orders that they are associated with. Furthermore, the application has to be structured in packages according to a layered architecture presented in the support presentation for this assignment.

For this assignment to be complete, it should use at least these following classes:

* Model classes that represent the data models of the application
* Business Logic classes that contain the application logic
* Presentation classes that are GUI related classes
* Data Access classes that contain and do the access to the database

# 2.Problem Analysis, modelling, scenarios and use cases

1.Problem Analysis

A database management system(DBMS) is a software package designed to define, manipulate, retrieve and manage data to and from a database. A DBMS generally is used to manipulate the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate the given data.

Usually, DBMS are set up on a specific data handling concept, as the practice of administrating a database evolves. The earliest databases only handled individual single pieces of specially formatted data. Today’s more evolved systems can handle different kinds of less formatted data and tie them together in much more elaborate ways.

2.Modelling

The goal during this assignment is for the user to be able to manage a database by inserting, editing, deleting and keeping track of all of the elements from the three tables: Clients, Products and Orders. The user has to be able to insert necessary data in textfields and using necessary buttons, he has to be able to perform the desired operations on the database.

3.Scenarios and use cases

A use case is a list of actions or event steps that define the possible interaction between the user(usually known in UML as an actor) and a system, in order to achieve their goal. The user can be in this case either a human or an external system. The use cases are in strong correlation with the steps that the user has to follow, this being one of the biggest reasons why I chose to make my GUI as friendly as possible.

a.Use cases

1.Use case: View all of the clients

Primary Actor: user

Resume: The user clicks the Client Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the View clients button, which presents to them a table containing all of the clients and their information, like name, age and email address.

2. Use case: Insert a client

Primary Actor: user

Resume: The user clicks the Client Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Add client button, which presents to them three textfields where the information of the newly to be added client will be specified. After the user has inserted all of the necessary information, he clicks the Insert client button, and the application inserts that user into the database

3. Use case: Delete a client

Primary Actor: user

Resume: The user clicks the Client Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Delete Client, which will present to them a drop down menu from which the user will choose the client to be deleted, then press the Delete selected button.

4.Use case: Edit a client

Primary Actor: user

Resume: The user clicks the Client Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Edit Client, which will present to them a drop down menu from which the user will choose the client to be edited, then the current information of the client will be shown, and the user will be able to make changes.

5.Use case: View all of the orders

Primary Actor: user

Resume: The user clicks the Order Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the View Orders button, which presents to them a table containing all of the orders and their information, clientid, productid, and quantity

6.Use case: Add order

Primary Actor: user

Resume: The user clicks the Order Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Add order button, which presents to two drop down menus to select the client and product and a text field in which to specify the quantity of the product for the order

7.Use case: Generate Bills

Primary Actor: user

Resume: The user clicks the Order Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Generate bills, which will create a PDF file for each order in the database

8.Use case: View all of the products

Primary Actor: user

Resume: The user clicks the Product Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the View products button, which presents to them a table containing all of the products and their information, like stock, name and price

2. Use case: Insert a product

Primary Actor: user

Resume: The user clicks the Product Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Add product button, which presents to them three textfields where the information of the newly to be added product will be specified. After the user has inserted all of the necessary information, he clicks the Insert product button, and the application inserts that product into the database

3. Use case: Delete a product

Primary Actor: user

Resume: The user clicks the Product Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Delete product, which will present to them a drop down menu from which the user will choose the product to be deleted, then press the Delete selected button.

4.Use case: Edit a product

Primary Actor: user

Resume: The user clicks the Product Menu button and is redirected to a screen containing all of the possible operations he can make. From here, he clicks the Edit product, which will present to them a drop down menu from which the user will choose the product to be edited, then the current information of the product will be shown, and the user will be able to make changes.

b.Use case diagram

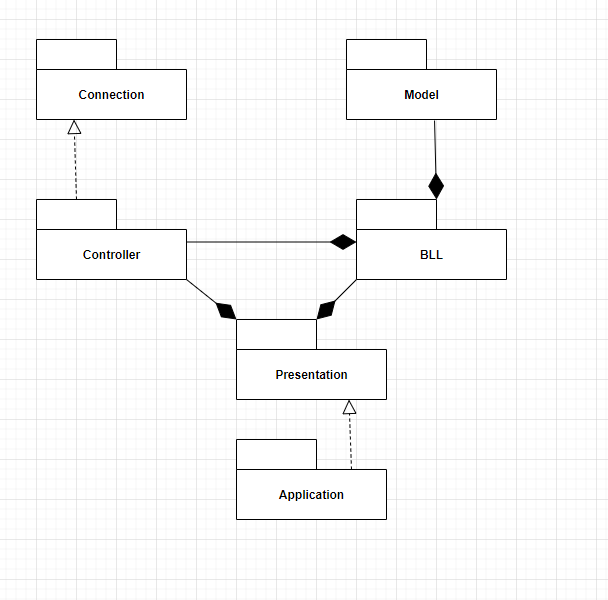
Diagram, schematic

Description automatically generated

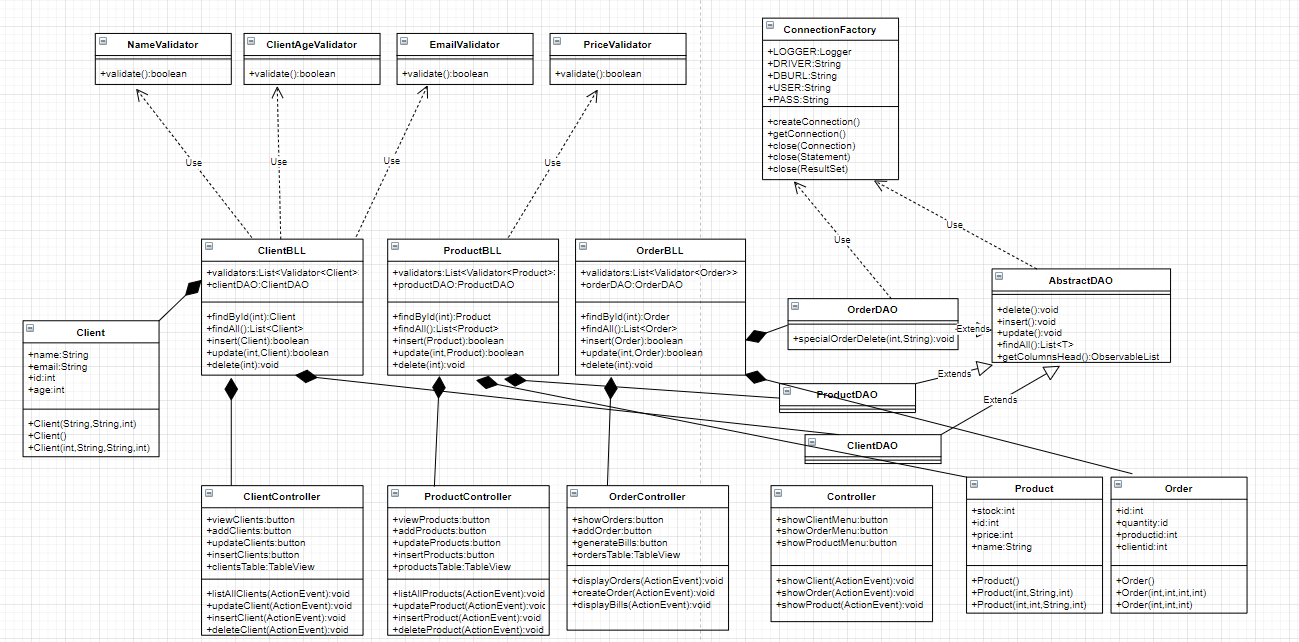
# 3.Design

1.UML Diagrams

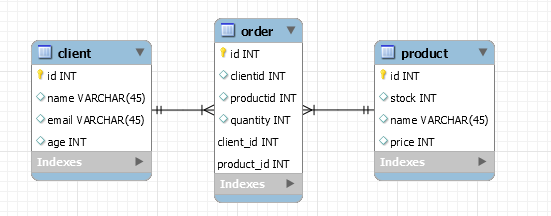
a.Package Diagram



b.Class Diagram



c.Database Schema



2.Data Structures

The data structures used in my program are mostly primitive data types such as integers and boolean, but I have also used some complex ones like ArrayList and the objects that I have created myself, such as: Client, Product and Order.

In order to be able to manage all the data concerning the three data types that I have created I have used the following classes, ClientDAO, ProductDAO and OrderDAO, all of which extend the class AbstractDAO, class which implements working with the database using reflection techniques to determine with which table the program has to work. This class implements all the necessary functions and methods for working with the database.

The ConnectionFactory class is the class that makes the connection between the project itself and the database created in MySQL.

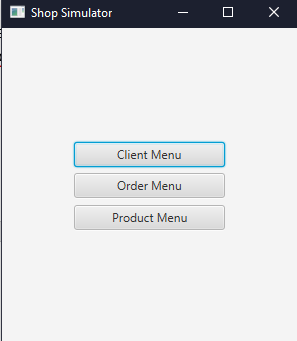
3.Packages

Java Packages are used in order to better organize modules and group classes and interfaces that are related in Object-Oriented Programming(OOP) development. For this project, as per the requirements, I have used a Layered Pattern Architecture

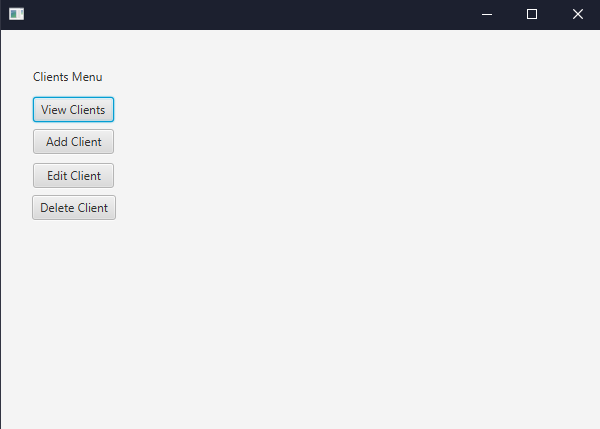
* BLL-contains the classes that check if the information supplied is correct and then executes the required querries for the respective table. I have three of these classes, ClientBLL, OrderBLL and ProductBLL, each of these being responsible to their respective Model methods
* Validators-contains the classes that validate the informations introduced for different fields of the application. ClientAgeValidator, EmailValidator, NameValidator, PriceValidator
* Connection-the ConnectionFactory is the class that makes the connection with the database, creating a statement. It also contains methods for closing this connection
* DAO-The AbstractDAO contains generic methods to insert into the database, delete from the database, update an entry from the database, prepare the Table for viewing. The classes ClientDAO, OrderDAO and ProductDAO extend this class and use the generic methods to implement operations on their respective table from the database.
* Model-contains the model of the data in the application, corresponding to the tables in the database. I have a Client, Order and Product class.
* Presentation-contains classes that are tasked with providing the user with a graphical interface that they can interact with. Each of the Menus has their own Controller and FXML class, each implementing different methods for realizing their respective functions. I have a ClientController, an OrderController and a ProductController, each with their own FXML file, which contains information about the placement of the GUI elements on the screen.

4.Graphical User Interfaces

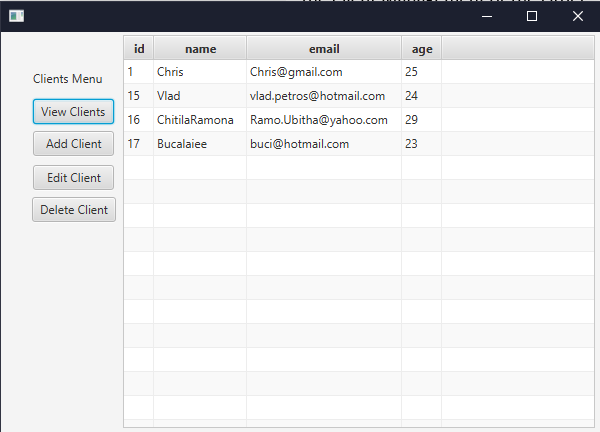
The Graphical User Interface represents the link between the software application that cannot be seen and the user, thus it has to be friendly and easy to use.



This is the main screen that appears when the user starts the application. From here, depending on the button that the user presses, he can navigate to the different parts of the application, such as the Client Management or the Order Management.



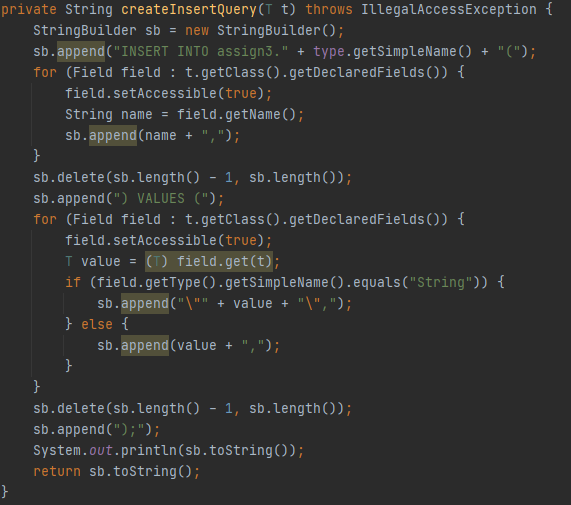
This is the Client Menu, where the user can choose to do a number of different actions, such as view all the clients, add a client or delete a client.

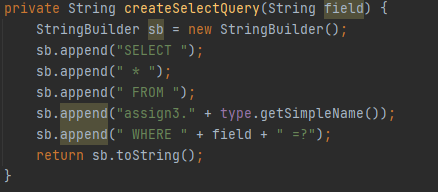
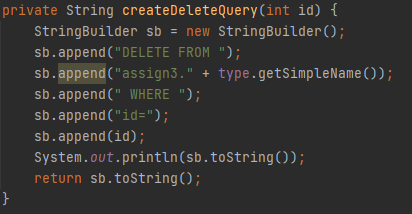


For example, if the user chooses he wants to see the clients currently in the database and clicks the View Clients button, a table will appear, displaying all the clients. The same action can be done with the Orders and Products Menus, each of them having a Display button, that will display a table containing the information from the database.

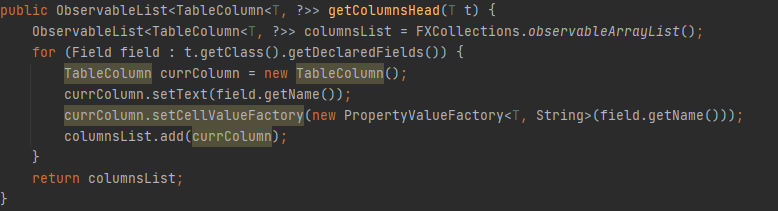
# 4.Implementation

Below are a few methods from the AbstractDAO class which implements the queries using reflection techniques.





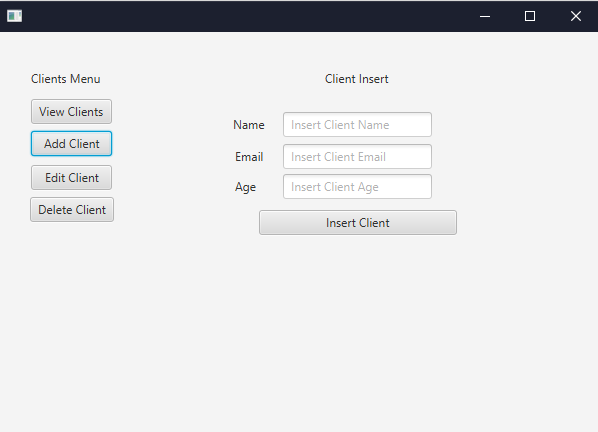
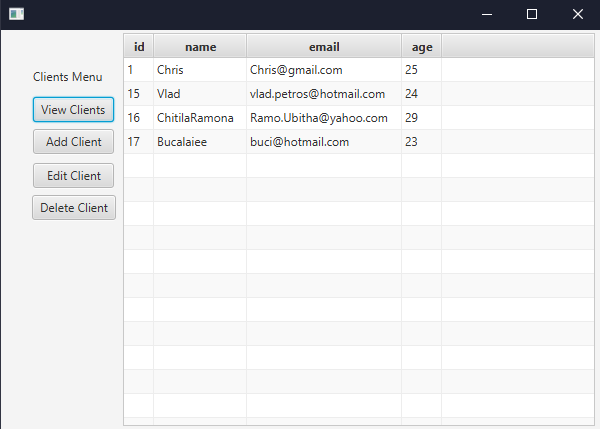
These methods return in the form of a string the query that has to be executed in order to achieve the desired function of the application.

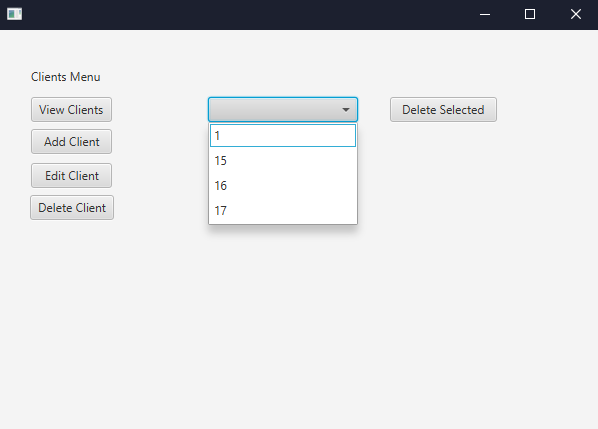
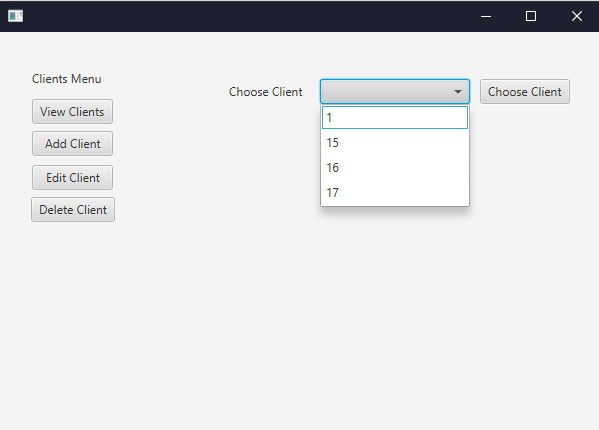


This method creates, using reflection techniques as well, the column headers of the table that will be used to display all of the Clients/Orders/Products in their respective Menus. This method does not populate the table with the values from the database, merely naming each column and setting its type(for example the first column will be named id and will have type int).

# 

# 5.Results





The pictures above show the evolution of the GUI when the user decides to perform different operations inside of the application.

# 6.Conclusions

This assignment has offered us a good opportunity to remember and apply the Object-Oriented Programming paradigms and concepts. It also offered us an insight into the process of designing and developing a software application.

Because I have never worked with a database from inside a Java application, I thought it would be difficult to understand and to work with queries from inside Java, but thanks to the Support Presentation and the support classes that were provided to us, it became quite easy to understand. Also, this was the first time I worked with reflection techniques and it was a great opportunity to understand what goes into building an application that does not rely on copy pasting code in several different places, and rather create a method that works in each case. I found this especially helpful, as I think it is important that we deal with a real life situation in order to more thoroughly understand the concepts that are presented to us.

What I can take from this assignment, as from all the others before it, is to not panic when I encounter errors and difficulties when designing a program, and rather stay calm and divide the problem into several smaller ones that are easier to manage, because otherwise such a project can quickly become overwhelming with the amount of things you have to consider when implementing methods. Another very important lesson is to always test your application, even if it could seem that you haven’t implemented a lot, as that helps avoid big errors in the long run.

I can say that I am really proud that I managed to power through this assignment through research and find solutions to all the problems that I’ve encountered along the way.

The project is by no means perfect or even close to being done. There is always room for improvement, such as adding more information about the Clients and the Products in the database tables, designing the GUI so it is even friendlier. Furthermore, the methods could always be improved to gain performance and their organization could bring a further ease of understanding them from the perspective of an outsider looking in.

# 7.Bibliography

* Programming Techniques – Lecture Slides by prof. Ioan Salomie
* Assignment 3 Support Presentation – Programming Techniques Teams
* <https://app.diagrams.net> – creating UML diagrams
* <https://www.techopedia.com/definition/24361/database-management-systems-dbms> - understanding the concept of a DBMS
* <https://www.baeldung.com/java-pdf-creation> - creating PDF files in Java
* <http://tutorials.jenkov.com/java-reflection/index.html> - reflection in Java