

# DOCUMENTATION

## ASSIGNMENT *ASSIGNMENT\_3*

STUDENT NAME: Sîrca Florentina Raluca  
GROUP: 30423

# CONTENTS

1. Assignment Objective .....	3
2. Problem Analysis, Modeling, Scenarios, Use Cases.....	3
3. Design .....	6
4. Implementation .....	8
5. Results.....	9
6. Conclusions.....	11
7. Bibliography .....	11

## 1. Assignment Objective

The objective of the assignment is to implement and design a warehouse management system with a graphical interface. The purpose is to enhance user experience and improve the data input process for clients, products, and orders. Additionally, the system should allow users to edit or delete existing items in the database. The management system should be user-friendly and easily understandable for users with varying levels of technological knowledge.

The side objectives considered when designing the main objective are as follows:

- Clearly define the main objective and sub-objectives required to achieve it, such as creating classes and packages for the computer system's composition.
- Analyze the problem and determine the functional and non-functional requirements. Develop graphical interfaces for adding, visualizing, and processing data.
- Design a solution that includes a parsing and storage method for user-inputted data.
- Establish a database connection for storing information about clients, products, and orders.
- Implement the solution and validate user-inputted data.
- Conduct testing to ensure the solution's functionality and effectiveness.

## 2. Problem Analysis, Modeling, Scenarios, Use Cases

SQL is a programming language used by almost all relational databases, for querying, managing and defining data, as well as for controlling the provision of access.

Initially developed by IBM in the 1970s with assistance from Oracle, SQL has evolved with contributions from companies like IBM, Oracle, and Microsoft. While it remains prevalent, SQL has also influenced the development of new programming languages.

Database software plays a crucial role in creating, editing, and maintaining files and database records. It simplifies tasks such as data entry, editing, updating, and reporting, while also managing data storage, backup, multi-access control, and security. Robust database security is particularly important due to the increasing occurrence of data theft. Database software is often referred to as a "database management system" (DBMS).

A DBMS serves as the interface between the database and end users or programs, enabling users to retrieve, update, and manage information efficiently. It facilitates supervising and controlling the database, allowing administrative operations such as performance monitoring, adjustment, backup, and recovery.

MySQL is an open-source relational DBMS that is highly optimized for web applications and can run on any platform. It has gained popularity among web developers and for web-based application development. MySQL excels at processing numerous queries and transactions, making it a preferred choice for e-commerce companies that handle substantial money transfers. Its flexibility is a key feature.

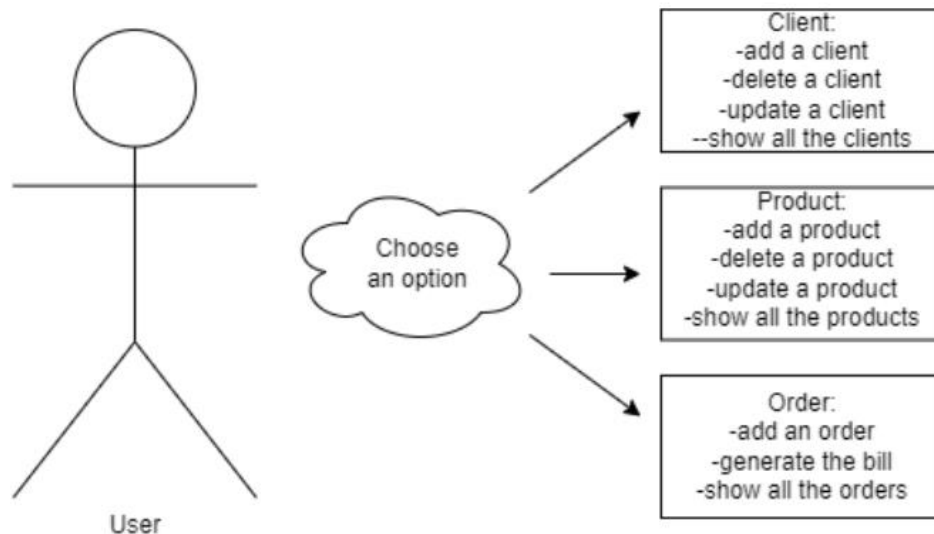
Reflection refers to a program's ability to analyze and process its own structure and source code. In Java, for example, a program can examine its classes, methods, and other details. Reflection allows programs to dynamically modify their structure, enabling tasks like updating packages while the program is running.

There are more others scenarios and use cases of the application, as shown above:

- Adding to a new line in the table (as well for the client and product).
- Updating a new line in the table.
- Deleting a line in the table.
- The view of the data from the database

The application design should adhere to a well-defined structure to ensure the correctness and security of processed data added or updated through the database. Various scenarios and use cases exist for the application, including adding new entries to tables (e.g., clients and products), updating existing entries, deleting entries, and viewing data from the database. Each case will have specified elements in the graphical user interface and implemented code to perform the respective actions.

## USE CASE



### a) Title: Add a Client

Resume: The user can add clients to the cart, by inputting the name, age and the address. The client will be added to the DataBase.

Actors: User

### b) Title: Delete Client

Resume: The user can remove a client he added to the table, by entering the client id of the client he wants to remove. The table will be updated without the client that he deleted. If the client he tries to remove is not in the table, an error message will be shown.

Actors: User

### c) Title: Update Client

Resume: The user can edit the details of an existing client. After entering the client id, he wants to edit, modify the data from the fields and clicking on the "Update" button. He can edit the following details: name, address, and age. If the client id is invalid, and error message will be displayed.

Actors: User

d) Title: Add Product

Resume: The user can add products to the cart, by inputting the product name, price, and the quantity he wants to order. The product will be added to the Data Base.

Actors: User

e) Title: Delete Product

Resume: Resume: The user can remove a product he added to the table, by entering the produc id of the product he wants to remove. The table will be updated without the product that he deleted. If the product he tries to remove is not the in the table, an error message will be shown.

Actors: User

f) Title: Update Product

Resume: The user can edit the details of an existing product. After entering the product id he wants to edit, modify the data from the fields and clicking on the "Update" button. He can edit the following details: name, price and stock. If the product id is invalid, and error message will be displayed.

Actors: User

g) Title: Show Product

Resume: The user can see all the products that are in the table only add products to the cart, by inputting the product id and the quantity he wants to order.

Actors: User

h) Title: Show Client

Resume: The user can see a table with all the clients just by pressing the "SHOW" button. He can see the id, name, age and address for each client.

Actors: User

i) Title: Show Order

Resume: The user can see a table with all the products just by pressing the "SHOW" button. He can see the id, name, age, and address for each product.

Actors: User

j) Title: Add an Order

Resume: The user can place an order after he inserts the client id, the product id, the quantity that he wants to buy from the shop. If the product id is invalid or the stock is less than the quantity that he needs, and error message will be displayed.

Actors: User

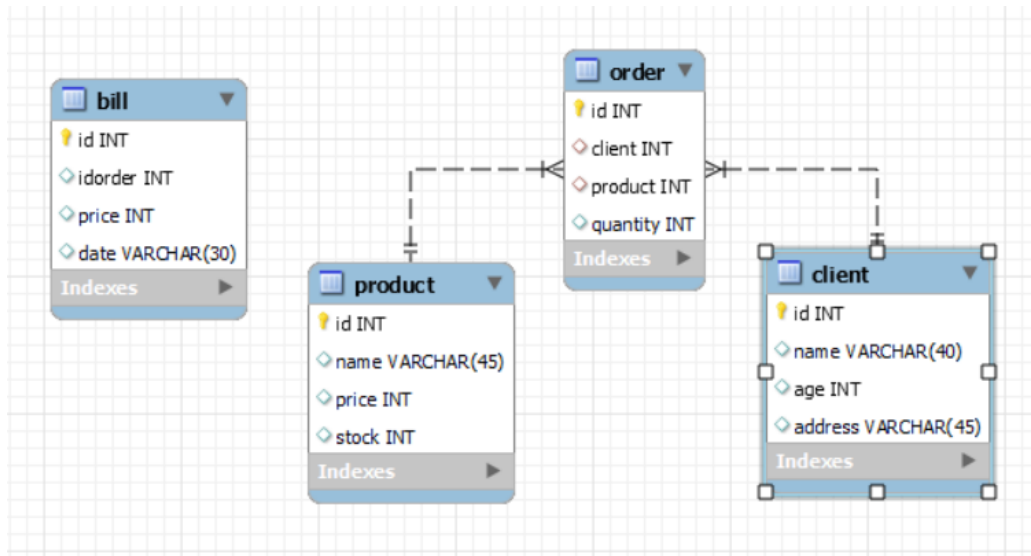
k) Title: Generate a bill

Resume: The user can generate a bill after he inserts the order, based on the product id, the quantity that he wants to buy from the shop and the date. If the product id is invalid, and error message will be displayed.

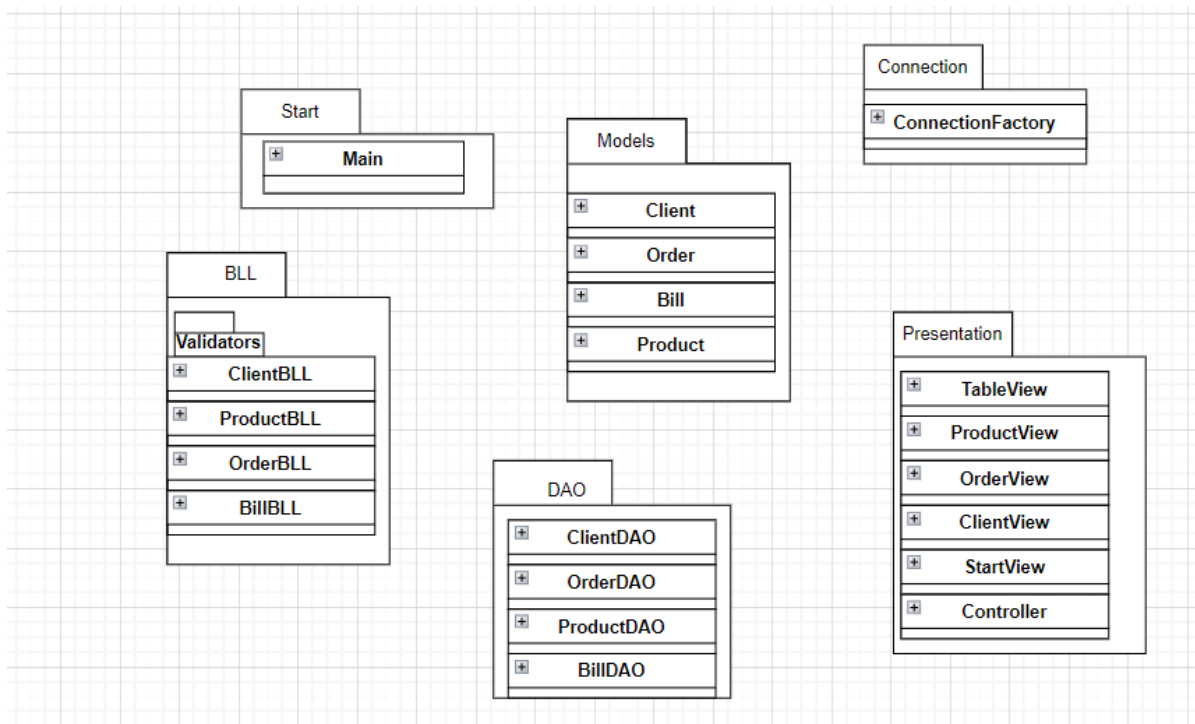
Actors: User

### 3. Design

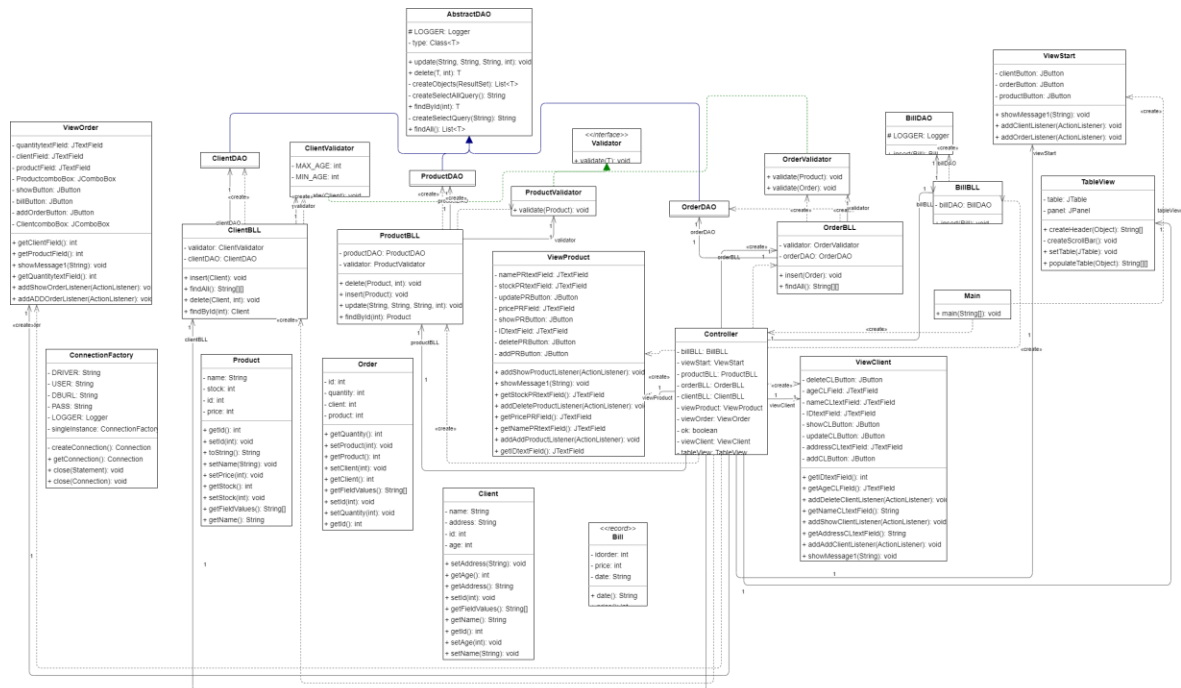
Database schema



Package Diagram



Class Diagram



The application contains more main classes which have the scope to define the application and the graphical user interface and the tables in which the information will be displayed in the database. For the design of this application the classes were organized by the model layered architecture which has at the base the information transmission

from a layer to another (packages and classes). This model makes the maintenance of the program easier and ensures the security of the data and their integrity by the fact that the package BussinesLogic, the data added by the user is verified before its update or to be added in the database.

For the data access is used the reflection technique, which is implemented in an abstract class which is extended in any other classes which uses a table type. So to add, to update or to delete some type of data there is no need for major modifications of the program. This method is used also for the table creation from the graphical user interface with the user. The reflection is used from the fact of generalization, because using this method we don't have to make modification to the data access logic of the data because the methods from the abstract class automatic extract the set and get methods populating them with current data. The data structures used are grouped in the Model layer of the project being implemented as elements of the database. These classes contain the exact same type of primitives variables which are accessed through reflection with the help of the methods get and set. In this package are stored the necessary methods to create the table structures in which will be displayed the data.

The graphical user interface is divided in many windows each with a different role. The main Interface (GUI) presents 3 buttons (Client, Product, Order) which when pressed will open a new window. The ClientGUI window will be generated by pressing the Client button and consists of object where the data itself will be stored as well as 4 buttons (Add Client, Update,Client, Delete Client, Show Clients) which will be used to process the data in the table.

We have chosen this implementation for an easier observation of changes made within the table. The ProductGUI window will be like the ClientGUI window, the only difference being that the table will display the information from the Product table. The OrderGUI window shows the Add button which will add a selected product to an order, this can be seen by clicking the View order button which will display all products added to the current order. Generate bill button will create a line in the table with the same name. The bills can only be inserted and read from the Log table; no updates are allowed.

## **4. Implementation**

In order to implement the model chosen for the application, we have divided the classes into several packages, depending on their functionality and role:

- connection: this package is used to establish a connection with the database implemented in MySQL Workbench using the ConnectionFactory class. In this class methods for creating the connection and closing the connection are implemented.

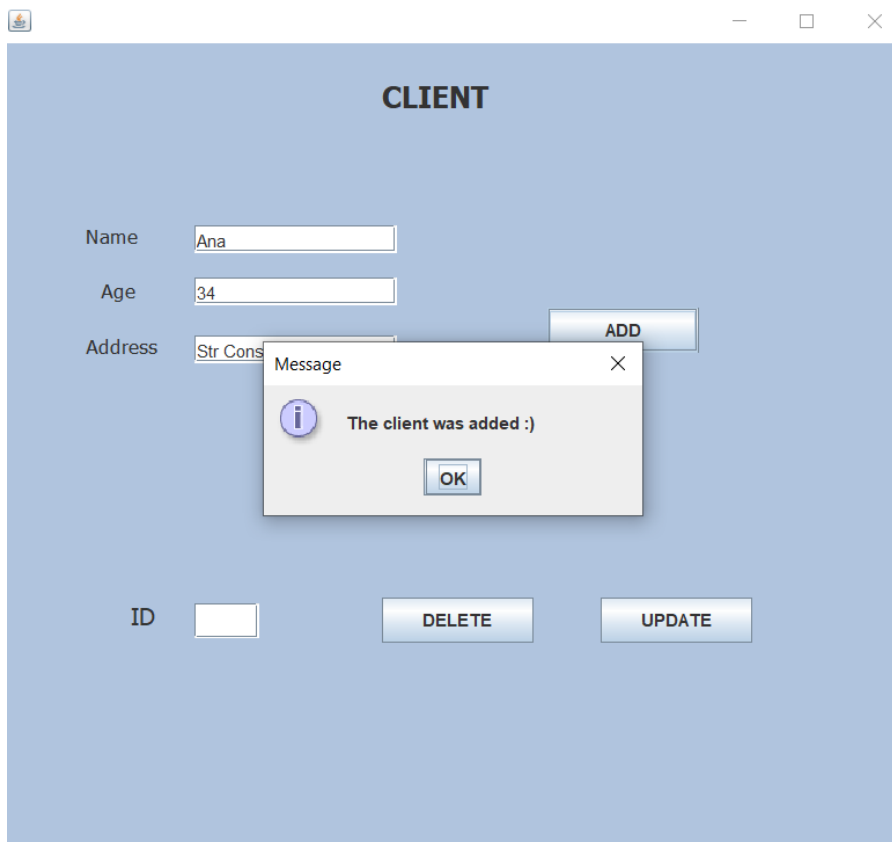



- dataAccessLayer: in this package an abstract class AbstractDAO is defined in which are implemented the basic methods for selecting, deleting, updating, and inserting new data into the database tables. These operations are performed by establishing them with the database using the package described above, and then using methods that return the object that fulfills the given requirements.

- businessLogic: Inside the methods in the classes of this package is the implementation of the application logic and how data is passed to the user through the GUI. Also, also within this package is implemented the validator new interface which will have the role of checking the data entered by the user and in case of a wrong entry will return a specific exception.

- view: this part of the application deals with retrieving data from the layer and processing it so that it takes a user-friendly form. In the case of this application, the data passed as a list of objects is entered in a table, and then displayed in a separate view. The user can use a variety of buttons to select the type of operation they wish to perform. After selecting an operation, the user can enter any values in the fields specific to each operation. Table creation is based on reflection, the constructor of the class responsible for this activity accepting a class from the "model" package from which it will extract the write methods and read methods ("getters" and "setters" ) and a list of objects of type model class from the which will extract the actual data.

## 5. Results





—

□

×

# ORDER

ID CLIENT

SHOW

ID PRODUCT


ADD

QUANTITY


generate bill

Message

×

 NEW BILL:PRODUCT 2, PRICE 102, DATE 18.5.2023

OK



—

□

×

# ORDER

ID CLIENT

SHOW

ID PRODUCT


ADD

QUANTITY

generate bill

Message

×

 order was added:)

OK

## 6. Conclusions

Throughout this project, I had the invaluable opportunity to expand my knowledge and proficiency in Java programming. One significant aspect that I delved into was establishing seamless connections to databases, which opened up new possibilities and functionalities for the system. It underscored the criticality of a well-organized layered data model, serving as the foundation for a robust and scalable architecture.

I discovered the immense value of mastering the effective utilization of Swing components, including JTable, JList, and more, throughout this project. This understanding empowered me to create highly interactive and user-friendly graphical interfaces. By the means of this project I managed to improve my knowledge about queues processing and how they are implemented on a system

In summary, this project served as a platform for deepening my understanding of Java programming and honing my skills in database connectivity. It further emphasized the significance of a well-structured layered data model, showcasing its myriad benefits.

## 7. Bibliography

1. <https://www.baeldung.com/java-jdbc>
2. <https://jenkov.com/tutorials/java-reflection/index.html>
3. <https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>
4. <https://www.baeldung.com/java-jdbc>