DOCUMENTATION

ASSIGNMENT *3*

STUDENT NAME: SOPONAR DARIUS

GROUP:30424

# CONTENTS

[1. Assignment Objective 3](#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](#_Toc128043140)

[3. Design 4](#_Toc128043141)

[4. Implementation 4](#_Toc128043142)

[5. Results 6](#_Toc128043143)

[6. Conclusions 6](#_Toc128043144)

[7. Bibliography 6](#_Toc128043145)

# Assignment Objective

The main objective of this assignment is to create an application which manages the order of

a warehouse. The clients, products are orders are stored in a database. The application is designed according to a layered architecture pattern.

# Problem Analysis, Modeling, Scenarios, Use Cases

The application is used to manage the clients, products and orders of a warehouse. There are

multiple operations can be done on the database: inserting (clients, products or orders), editing (clients or products), deleting (clients or products) or viewing (clients, products or orders).

A screenshot of a computer screen

Description automatically generated with low confidence

Inserting a new client means that the user is allowed to enter a new client including his ID, email, name, age and address. If inserting a new product you will need an item’s ID, it’s name, quantity and price. The edit operation allows the user to modify the database. The delete operation simply deletes a client or item from the database by its ID. The view operation will show the table of the available clients, products or orders.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer error message

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

# Design

The project is structured in a layered architecture pattern, existing more packages which contain the classes. The presentation layer, which contains classes defining the user interface, the business layer which contains the classes that encapsulate the application logic and the data access layer which contains the query and the database connection.

Here is the UML diagram of the project:

A screenshot of a computer program

Description automatically generated with medium confidence

# Implementation

There are 3 classes called “ClientAge”, “ProductPrice” and “ProductStock” which verify if the adding, deleting or editing operations make logically sense. For example, when wanting to insert a new product it’s price would have to be between 1 and 10000, any other value chosen resulting in a message such “The product price limit is not respected”.

The “ClientBLL” class is responsible for managing client-related operations. It utilizes a ”ClientDAO” for data access and includes validators, such as “ClientAge”, to enforce validation rules. The “ClientBLL” class provides methods to find clients by ID, retrieve all clients, find clients by name, insert a new client, update an existing client, and delete a client by ID. It performs validation using the registered validators before executing insertion or update operations. Overall, the “ClientBLL” acts as a bridge between the presentation layer and data access layer, encapsulating the business logic and validation rules for client operations.

The “OrderBLL” class responsible for managing order-related operations. It interacts with a “OrderDAO” for data access and has dependencies on “ProductBLL” and “ClientBLL” classes.

The “OrderBLL” class provides methods to find orders by ID, retrieve all orders, and insert new orders. The “insertOrder” method performs validations, checks if the associated client and product exist, and verifies sufficient stock for the order. It then inserts the order, updates the product's stock, and handles product deletion if the stock reaches zero.

The “ProductBLL” class responsible for managing product-related operations. It interacts with a “ProductDAO” for data access and includes validators for product validation. The “ProductBLL” class provides methods to find products by ID, retrieve all products, find products by name, insert a new product, update an existing product, and delete a product by ID. Validation is performed before insertion or update operations using the registered validators. If any validation rule is violated, an exception is thrown.

The “AbstractDAO” class is a generic base class for Data Access Object (DAO) classes. It provides common database operations such as inserting, updating, deleting, and retrieving objects from the database. Using reflection, it dynamically generates SQL queries based on the object's properties. The class includes methods for finding objects by ID or name, retrieving all objects, and executing the generated queries. With the help of the “ConnectionFactory” class, it manages database connections and handles exceptions related to database operations. The Client, Product and Order DAO classes will extend the AbstractDAO class.

The `deletequery` class enables dynamic construction of SQL delete queries. It provides methods to set the table name, field name, and value for the delete operation. The `build` method constructs the query string using the provided components, following the syntax "DELETE FROM [table] WHERE [field] = [value]". The `clear` method resets the query components. This class facilitates flexible deletion operations in the database access layer.

The `Client` class represents a client entity with properties such as ID, name, email, address, and age. It provides constructors to initialize the object with these properties. The class also includes getters and setters for accessing and modifying the property values. This class serves as a model for storing and manipulating client information within an application. The same logic applies to the ‘Order’ and ‘Product’ classes.

The code you provided is a Java program that creates a simple graphical user interface (GUI) for managing orders, clients, and products. It uses Swing, a Java GUI toolkit, to create the interface components such as buttons, labels, and text fields.

The “Start” class represents the entry point of the program. It creates a JFrame and adds three buttons for "Product," "Client," and "Order" windows. Each button has an ActionListener that opens a specific window when clicked. The `clientWindow`, `productWindow`, and `orderWindow` methods are responsible for creating and displaying the windows for managing clients, products, and orders, respectively. Each window contains buttons for inserting, editing, deleting, and viewing the corresponding data. The `createClientWindow`, `editClientWindow`, `deleteClientWindow`, and `viewClientWindow` methods handle the actions performed in the client window. They create new frames for inserting, editing, deleting, and viewing client data, respectively. Similarly, the `createProductWindow`, `editProductWindow`, `deleteProductWindow`, and `viewProductWindow` methods handle the actions in the product window. The `createOrderWindow` method handles the actions in the order window. It creates a frame for inserting order data. The `createAndShowTable` method (not included in the provided code) is likely a helper method used to display data in a table format.

# Results

The updated results after any operation can bee seen in the database.

# Conclusions

Doing this assignment I have learned to organize the classes better and I have also reminded myself the efficiency of a database.

# Bibliography

<https://stackoverflow.com/>

<http://tutorials.jenkov.com/java-reflection/index.html>

http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/