DOCUMENTATION

ASSIGNMENT *3*

STUDENT NAME: Urdea Mara-Cristina
GROUP: 30422

CONTENTS

1. Assignment Objective

Consider an application Orders Management for processing client orders for a warehouse. Relational databases should be used to store the products, the clients, and the orders. The application should be designed according to the layered architecture pattern and should use (minimally) the following classes:

• Model classes - represent the data models of the application
• Business Logic classes - contain the application logic
• Presentation classes – GUI related classes
• Data access classes - classes that contain the access to the database

2. Problem Analysis, Modeling, Scenarios, Use Cases

Problem Analysis
A warehouse management system aims to streamline and automate the processes involved in managing inventory, tracking orders and client database. The system should provide functionalities such as inventory management, order processing and reporting.

Modelling the problem
To model the Warehouse Management System, we need to define the entities, events, and processes involved in the system. This includes modeling the inventory, orders, reports and clients. The inventory model, aka products model encompasses items names, their IDs, and quantities. The order model handles customer orders, their IDs and quantities ordered. The client model handles the data for clients like ID, name, phone, address and email. The modeling process will consider the interactions and dependencies between these entities and processes, ensuring an efficient and effective warehouse management system.

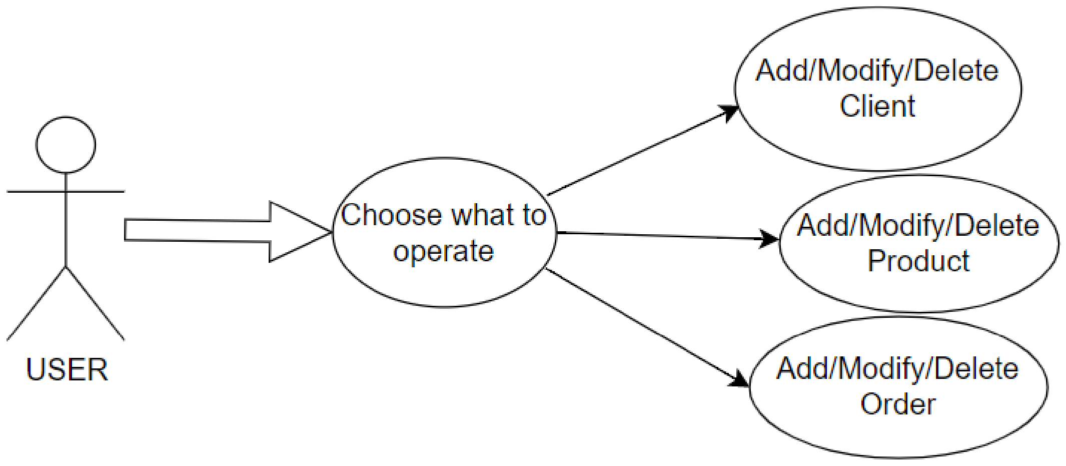
Use cases and scenario
The primary use case for the Warehouse Management System is to streamline and automate warehouse operations, including inventory management, order processing, reporting and clients management.
The user can manage inventory by adding, removing, and updating items, ensuring accurate stock levels. When processing customer orders, the system retrieves order details, checks inventory availability, and updates quantities accordingly.

3. Design

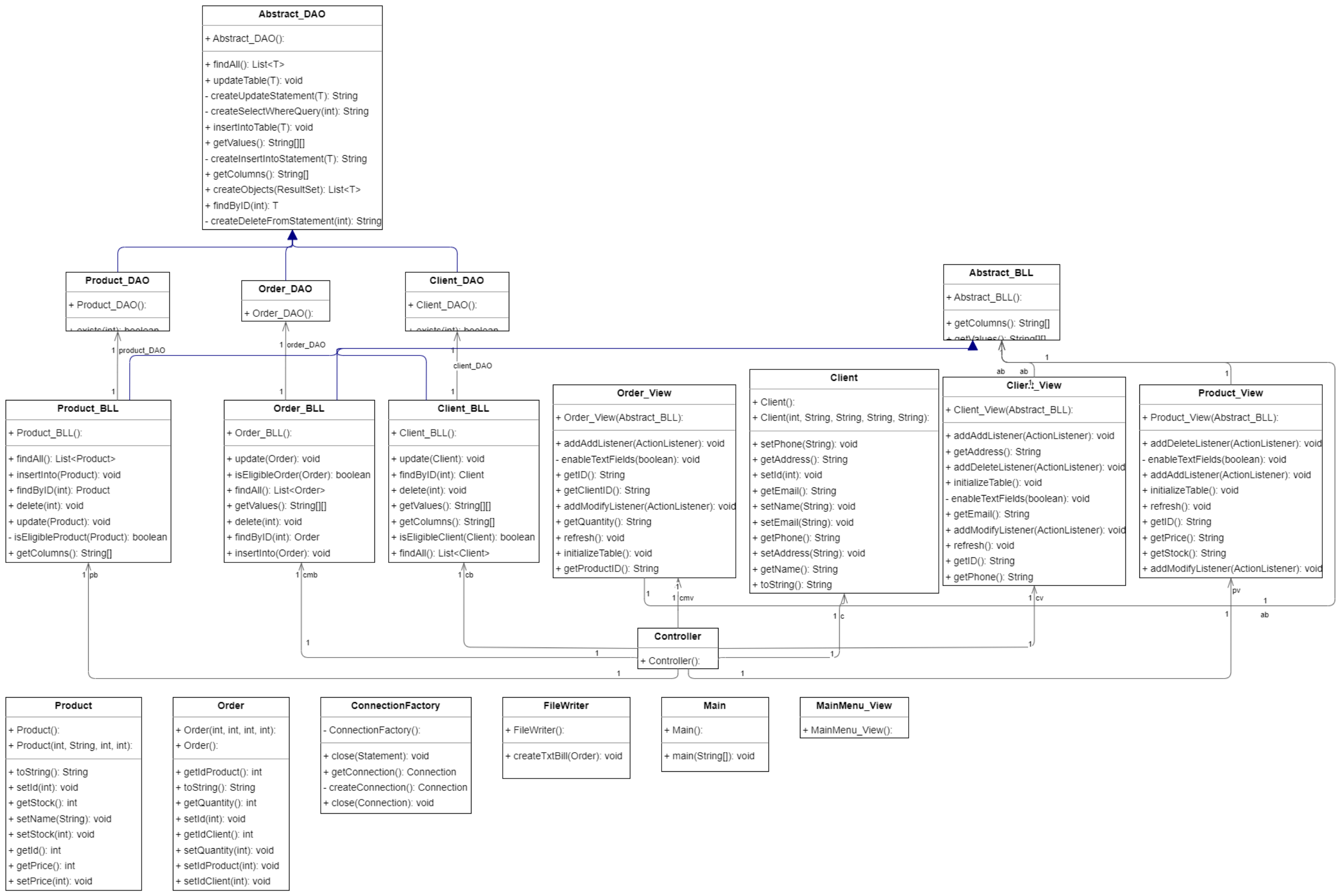
The Warehouse Management System follows an object-oriented (OOP) design approach, leveraging the principles of encapsulation, inheritance, and polymorphism. The system is designed to have modular components that can be easily maintained and extended. The application is divided into several packages, each representing a different aspect of the application’s functionality, such as user interface, clients management, products management, orders management. The class diagram shows the relationships between the different classes, such as the Client class, Order class, Abstract_DAO, Abstract_BLL etc.

The application's user interface is designed to be intuitive and easy to use. The user can input the necessary data, in the corresponding window, such as the client data, the product details and quantity and order details. The application then manages the database accordingly.

User diagram

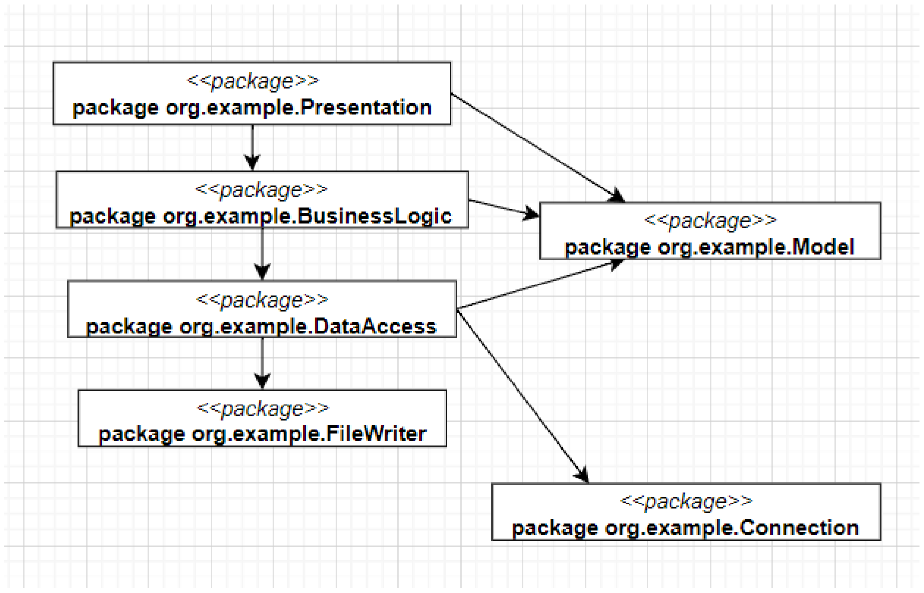


Class diagram



The UML, Unified Modelling Language, diagram presented above is a visual representation used to illustrate the software system. It includes the Java classes used to implement the project and their relationships and dependencies.

Package diagram
The purpose of Java packages is to group related classes together for easy organization. In Object-Oriented Programming (OOP), the Model-View-Controller (MVC) pattern is a common approach to designing projects for optimal efficiency. This pattern is widely used in most of the programming languages.


<<package>> package org.example.Presentation
<<package>> package org.example.BusinessLogic
<<package>> package org.example.Model
<<package>> package org.example.DataAccess
<<package>> package org.example.FileWriter
<<package>> package org.example.Connection

4. Implementation

'Client' class:

- It contains information about the clients, their ID, name, email, address and phone.
- Has the basic constructor, getters and setters
- toString() function used for displaying in logs/interface.

'Product' class:

- It contains information about the product, their ID, name, price and stock
- Has the basic constructor, getters and setters
- toString() function used for displaying in logs/interface.

'Order' class:

- It contains information about the orders, their ID, quantity ordered, the ID of the product wanted by the client and the ID of the client that orders.
- Has the basic constructor, getters and setters
- toString() function used for displaying in logs/interface.

‘Client_View’, ‘Product_View’, ‘Order_View’ classes:

- Includes all the elements that make up the JFrame displayed on the screen (buttons, labels, text fields).
- I created a JPanel in which I arranged all the elements, and then add it to the JFrame.
- It has all the methods that add ActionListeners for each button

‘Abstract_DAO’ class:

- This class was created to define the common operations for accesing tables
- I have methods to create queries in order to perform CRUD operations
- I have a method to create the JTable

‘Client_DAO’, ‘Product_DAO’, ‘Orders_DAO’ classes:

- All of them have the same functions, for CRUD operations. All this functions come from the ‘Abstract_DAO’, but they are specifically implemented for said classes.

‘Client_BLL’, ‘Product_BLL’, ‘Orders_BLL’ classes:

- They have the same functionalities, calling the methods in the DAOs for creating the tables, updating the object chosen, deleting the object chosen and inserting the object chosen after their validation.

‘Controller class:

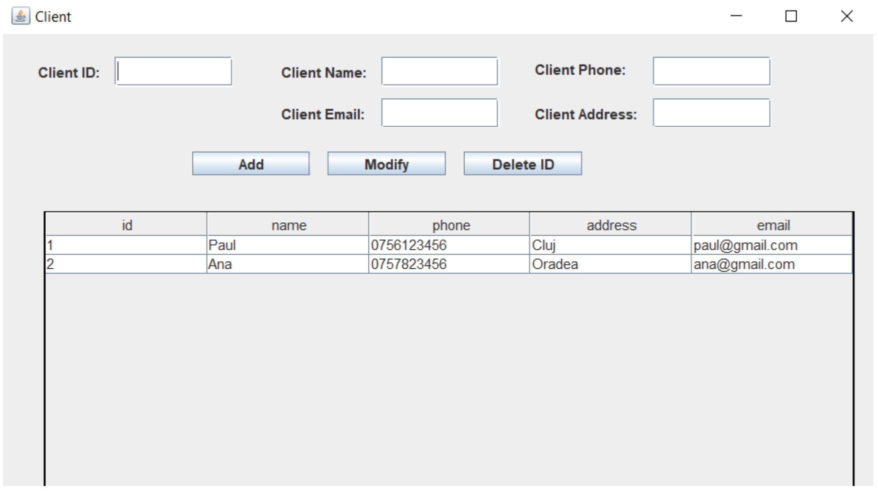
- In here action listeners for the views are implemented.

‘ConnectionFactory’ class:

- In here the connection with the Postgre database is made.

5. Results

For the results, the app was tested. Here you can see a few insertions that were made for clients. The same can be done with the others.



id	name	phone	address	email
1	Paul	0756123456	Cluj	paul@gmail.com
2	Ana	0757823456	Oradea	ana@gmail.com

6. Conclusions

In conclusion, the development of the Warehouse Management System has been a valuable experience for me. It has provided me with a deeper understanding of various aspects of software design and implementation. Through this project, I have gained proficiency in handling database interactions, implementing DAO classes, and establishing connections using the connection factory.

Throughout the development process, I have encountered and overcome challenges, requiring me to conduct research and troubleshoot issues independently. This hands-on problem-solving has significantly enhanced my understanding of various concepts and sharpened my ability to tackle similar challenges in the future.

Overall, this project has been instrumental in deepening my knowledge and skills in software development. It has provided a solid foundation for further growth and exploration in the field of application development.

7. Bibliography

https://dsrl.eu/
https://gitlab.com/utcn_dsrl/pt-reflection-example
https://www.postgresql.org/
https://www.geeksforgeeks.org/
https://www.baeldung.com/javadoc