DOCUMENTATION

ASSIGNMENT *3*

STUDENT NAME: Gengiu Robert Constantin
GROUP: 30421

CONTENTS

1. Assignment Objective

The main objective of the project is to create an application for implementing a customer, product, and order management system within a warehouse. Relational databases are used to store the products, customers, and orders. Additionally, the application should be structured into packages using a layered architecture. It should follow at least the following classes:

Model classes - the data models of the application;

Business logic classes - implement the application's logic;

Presentation classes - implement user input/output;

Data access classes - implement database access.

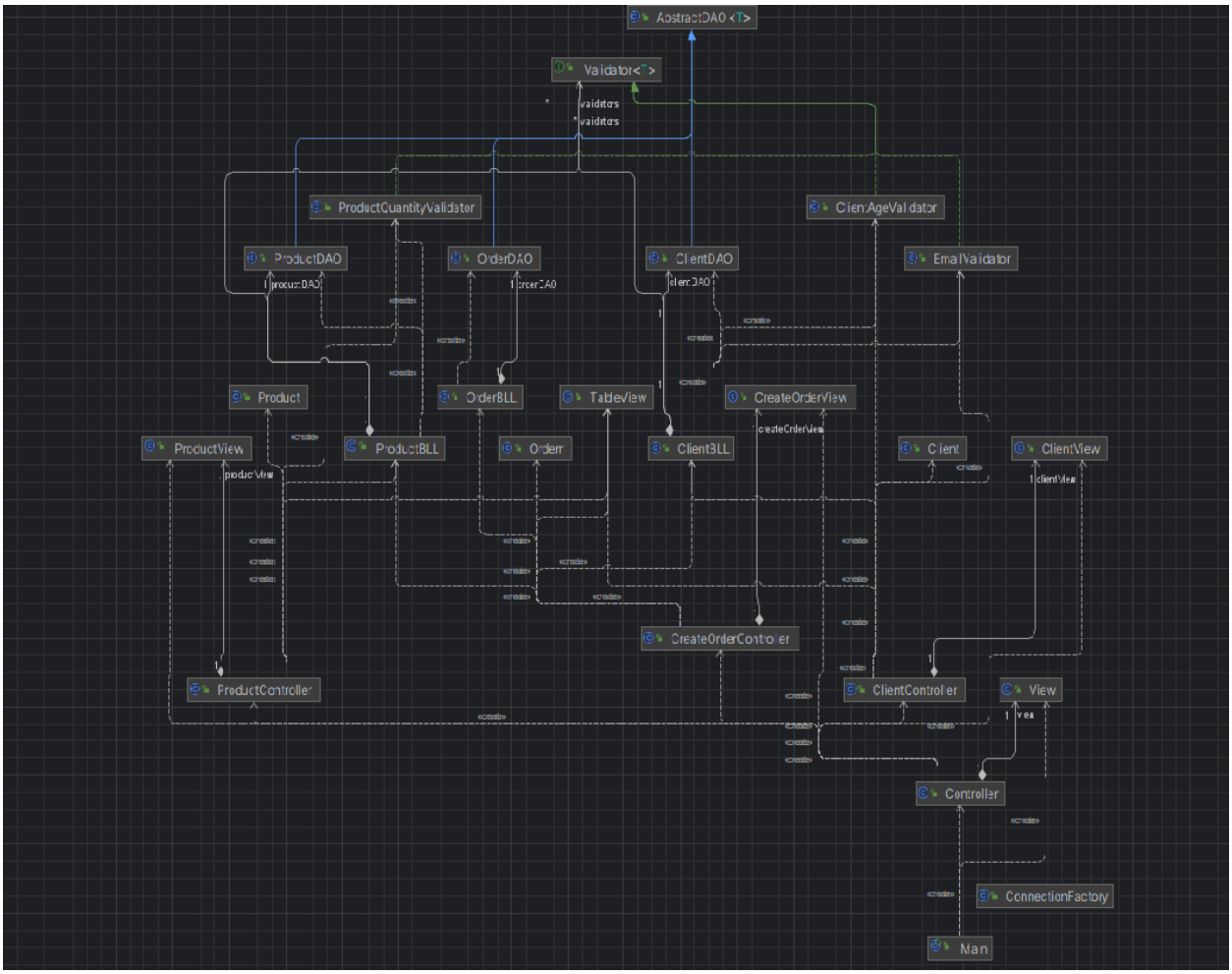
2. Problem Analysis, Modeling, Scenarios, Use Cases

The program's usage involves enabling CRUD (Create, Read, Update, Delete) operations on both the products in the warehouse and the customers of the company managing the warehouse. Additionally, it should provide a graphical interface for the three types of objects and implement their specific functionalities.

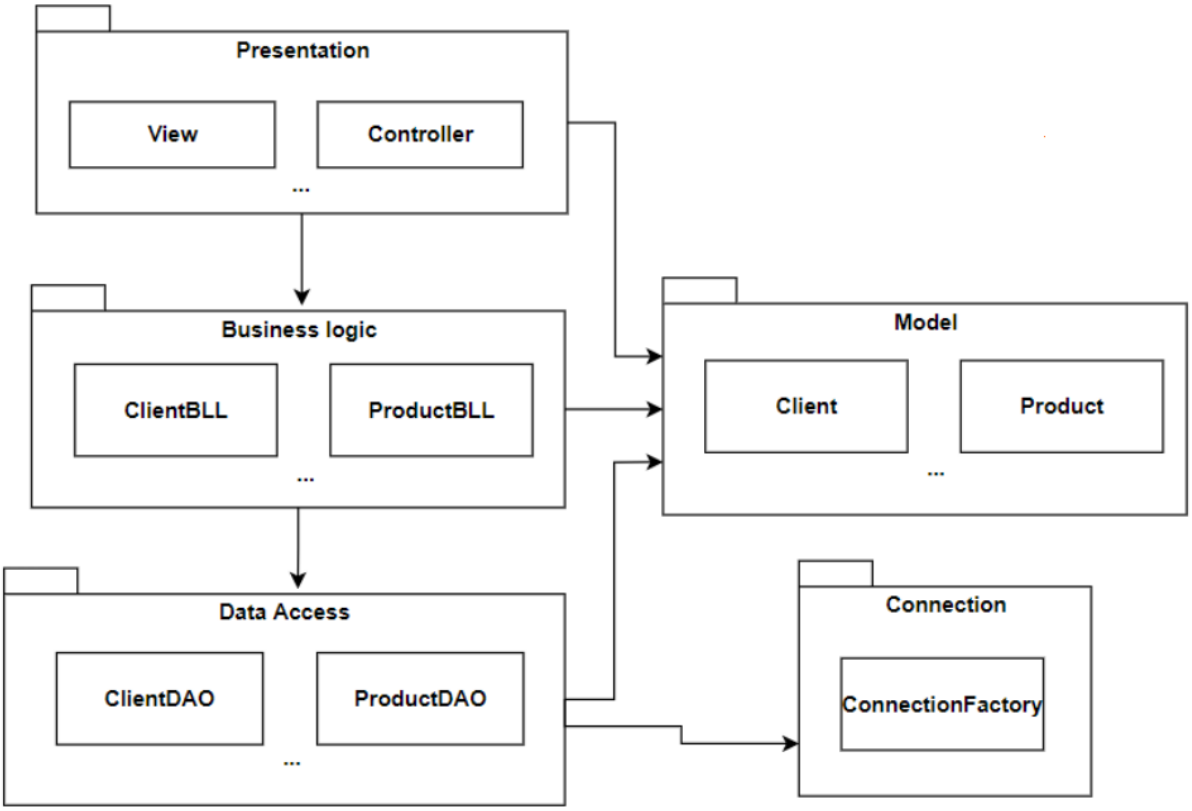
While CRUD operations are performed only on Products and Customers, within the Order, upon pressing the add button, an invoice with details about it will also be generated.

3. Design

Class Diagram



Package Diagram



4. Implementation

The main objective of the assignment is to create an application for implementing a customer, product, and order management system within a warehouse. Relational databases are used to store the products, customers, and orders. Additionally, the application should be structured into packages using a layered architecture. It should include the following classes:

Model classes: Data models of the application.
Business Logic classes: Implement the application logic.
Presentation classes: Handle user input/output (GUI).
Data Access classes: Implement data access to the database.
Connection package with ConnectionFactory class: Responsible for establishing a connection between the Java application and the MySQL relational database.
Start package with Start class: Links the three basic components of the Model-View-Controller architecture and contains the main method.
Notable algorithms:

To properly manage the client, product, and order tables, reflection techniques are used. Each package contains specific classes:

BusinessLogic package:
Validator classes: Implement validation logic for clients' age, email, and product quantity.
BLL classes: Contain a list of validators and a Data Access Object (DAO) for each entity (Client, Order, Product) responsible for database operations.

DataAccess package:
AbstractDAO class and its subclasses (ClientDAO, OrderDAO, ProductDAO): Implement Data Access Object (DAO) pattern using reflection techniques.
Presentation package:
View classes: Responsible for GUI design, including buttons, text fields, labels, and dropdown menus.
Controller classes: Establish a connection between Model and View without them having direct knowledge of each other. Contains listeners for GUI elements.
Model package:
Contains classes representing tables in the database: Client, Order, and Product.
Main package:
Contains the Main class, which initializes the Welcome view and its corresponding controller.
The package division outlined above ensures proper organization and separation of concerns within the application..

5. Results

During application runtime, the system demonstrates seamless functionality, allowing users to efficiently manage client, product, and order data within the warehouse environment. Key outcomes include:

Efficient Data Management: Users can perform CRUD operations effortlessly, ensuring smooth handling of client information, product inventory, and order processing. The intuitive interface simplifies navigation and interaction, enhancing user productivity.
Real-time Updates: The application provides real-time updates on inventory levels, client details, and order status, enabling users to make informed decisions promptly. This ensures transparency and accuracy in warehouse operations.
Invoice Generation: Upon order placement, the system automatically generates detailed invoices, providing comprehensive information about the transaction. This feature streamlines billing processes and enhances customer service.
Robust Architecture: The application architecture, based on the MVC pattern, promotes modularity, scalability, and maintainability. The segregation of components into distinct layers facilitates code organization and future enhancements.
Data Integrity: Through the use of DAO patterns and reflection techniques, the application ensures data integrity and security. Database operations are performed efficiently, minimizing the risk of data corruption or unauthorized access.
Enhanced User Experience: With a user-friendly interface and responsive design, the application delivers an enhanced user experience. Users can navigate through various features effortlessly, improving overall satisfaction and usability.
Overall, the application achieves its objectives of facilitating warehouse management operations while adhering to best practices in software design and development.

6. Conclusions

Through the completion of this assignment, I gained valuable insights into several key concepts.

Reflection: I acquired a deeper understanding of reflection and its applications in Java programming. The ability to examine and modify the structure and behavior of classes at runtime proved to be particularly useful, especially when working with dynamically changing data.
Data Access Object (DAO): I familiarized myself with the DAO pattern and its significance in separating business logic from data access code. Implementing DAOs facilitated the interaction with relational databases, allowing for efficient CRUD operations on objects stored in the database.
In terms of future development, several enhancements could be considered:

Enhanced User Interface: While the current application provides functional GUI elements, further improvements could be made to enhance the visual aesthetics and user experience. Incorporating more modern design principles and interactive elements could make the application more engaging.
Data Validation: Implementing additional data validation mechanisms and type checking would enhance the robustness of the application. Providing user-friendly error messages or warnings for incorrect input would improve usability and prevent data inconsistencies.
Order Management: To improve order tracking and management, incorporating features such as maintaining a global order count and including this information in invoice generation could provide valuable insights into order history. Additionally, optimizing the efficiency of order processing could be explored further.
Invoice Generation: Consideration could be given to generating invoices that consolidate all products ordered by a single client into a comprehensive invoice. This would provide a clearer overview of the client's purchases and streamline billing processes.
Overall, this assignment provided valuable hands-on experience and laid the foundation for further exploration and refinement of software development skills, particularly in the realm of database interaction and user interface design.

7. Bibliography

1. *https://dsrl.eu/courses/pt/*

2. https://www.baeldung.com/java-dao-pattern
3. https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html
4. https://www.mysql.com
5. https://en.wikipedia.org/wiki/Parsing