**DOCUMENTATION**

**ASSIGNMENT *3***

STUDENT NAME: Hoza Violeta Maria

GROUP: 30424

**CONTENTS**

[1. Assignment Objective 3](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043140)

[3. Design 4](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043141)

[4. Implementation 6](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043142)

[5. Results 8](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043143)

[6. Conclusions 9](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043144)

[7. Bibliography 9](file:///C:\Users\Violeta\Desktop\facultate\an2\sem2\PT\PT2024_30424_Hoza_Violeta_Maria\PT2024_30424_Hoza_Violeta_Maria_Assignment_1\Documentation1.docx#_Toc128043145)

1. **Assignment Objective**
2. ***Main Objective***:

The main objective for this project is to design and implement an order management application for the client orders for a warehouse. The user has a couple of options, for example to add, delete, edit, and view a specific client or product and even to create an order. The application will store the client, product, and order data in the database. Some basic information about the clients, products and the orders are stored. Not all orders will be accepted, a filter will be implemented such that orders that do not match the criteria will not be accepted. This filter consists of checking if the warehouse has at least as many products of the type of product that is being ordered.

1. ***Sub-objectives:***

* Analyze the problem and identify requirements.
* Design the orders management application.
* Implement the orders management application.
* Create a bill file that contains information about each order.
* Use a layered architecture.
* Create generic methods.
* Test the orders management application.

1. **Problem Analysis, Modeling, Scenarios, Use Cases**
2. ***Functional requirements***

* Must have: A connection to a database which stores all the relevant information about each client, product, and order. A graphical user interface that allows the user to interact with the application and the database implicitly. CRUD operations on the database.
* Should have: Generic classes and methods that allow the use of the same methods on different types of objects, without duplicate code.
* Could have:
* Won’t have: Login page for client/employee. Lack of ability for the client to perform only client operations makes the app useable only by the employees.

1. ***Non-functional requirements:***

* Ease of use: the user interface should be simple and intuitive, with labels that indicate what the application is expecting from the user.
* Compact design: the application’s graphical interface should not have a greater size than necessary but should be big enough to display all the information that has to be shown.

1. ***Use cases:*** The employee can choose between client or product operations or create an order.

* For the first two options, the user can: add a new client / product, delete them from the database, edit or update them. When the option view object is chosen, a table with all the entries appears.
* For the create order operation, with the help of two JComboBox, one for clients and the other one for the products, the employee will have the ability to choose an option (a client/product from the list). Another field must be completed where the user is asked for the quantity the user wants to buy. The submit option will finish the order and save it into the database.

A diagram of a diagram

Description automatically generated

1. **Design**

* Package diagram:

The application is divided into several packages, each containing related classes. These packages include Application, Presentation, Model, BusinessLogic, Validators, Connection and DataAccess.

A diagram of a computer

Description automatically generated

There are four layers in this architecture:

• The presentation layer – is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. In the current application, it receives which is the next step the user wants to take and the data input (if it is the case). It also displays new windows bringing the information the user wanted.

• Business logic - handles the business rules, calculations, and logic within an application which dictate how it behaves. It performs data validations and decides which will be the next step considering the input from the user. It also communicates with the database in the sense that it uses the methods form the data access layer to perform specific operations on the database.

• Data Access – contains methods for accessing the underlying database data. It implements CRUD operations and supports any class due to the generic implementation of the methods. Another important aspect is the connection between the application and the database. It uses a singleton class that makes sure that only one connection is made to the same database.

• Model – this is where all the data structures are implemented. There is a class for clients, products, and orders respectively.

* Class Diagram:

A computer screen shot of a diagram

Description automatically generated

1. **Classes Implementation**

* Application: The Main class is used to start the application.
* Model

Client – This class represents a client entity in the system. It contains fields for the client's unique identifier, age, name, address, and email address. The class provides constructors to initialize a client object, as well as getter and setter methods for accessing and modifying the client's properties. Additionally, it overrides the toString() method to provide a string representation of the client object.

Product - This class represents a product entity in the system. It contains fields for the product's unique identifier, name, price, and quantity available in stock. The class provides constructors to initialize a product object, as well as getter and setter methods for accessing and modifying the product's properties. Additionally, it overrides the toString() method to provide a string representation of the product object.

Order - This class represents an order entity in the system. It contains fields for the order's unique identifier, client ID, product ID, and quantity. The class provides a constructor to initialize an order object with the given client ID, product ID, and quantity, and it generates a random order ID within a specified range. Getter methods are provided to access the order's properties.

Bill - This record represents a bill generated for a purchase in the system. It includes fields for the order ID, client information, product information, total amount, and timestamp of the bill. It serves as a convenient and immutable way to store bill data.

Log - This class represents a utility for logging bill information to a file. It provides methods for appending bill information to a log file and formatting bill data into a string. The log file path is specified as a constant. The appendBillToFile method appends bill information to the log file, while the formatBill method formats bill data into a readable string format.

* Presentation
* BusinessLogic

ClientBLL -

ProductBLL -

OrderBLL -

* Connection

ConnectionFactory - This class manages the database connection for the application. It provides methods for obtaining a connection to the database, closing the connection, statement, and result set, and executing SQL queries. It uses the JDBC driver to establish the connection to the MySQL database specified by the DBURL constant. The class also handles exceptions related to database connectivity.

* DataAccess

AbstractDAO – This class implements methods that allow CRUD operations on the database. This class contains generic methods that also use reflection techniques such that the same code of the same method is used to perform same operation on the database, but any class is accepted. Reflection techniques assure extractions of the values for each field of the given object and also the names and type of those fields. This technique brings reusability to the code since it can be used for any object type. The methods implemented in this class are used in the controller package to communicate with the database and perform the tasks requested by the user.

1. **Results**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. **Conclusions**

This project helped me discover new functionalities in Java programming language, for instance, how to use reflection techniques, generic classes, and methods and about the layered architecture, which creates a cleaner and more decoupled code. Moreover, I learned about MySQL Workbench and how to use it. All the main and secondary requirements have been accomplished. At the end of the assignment, I consider that my Java programming skills evolved since the last project.

1. **Bibliography**
2. [*www.tutorialspoint.com*](http://www.tutorialspoint.com)
3. *Object-Oriented Programming - Lecture Slides of prof. Marius Joldoș*
4. *Programming Techniques - Lectures of prof. Cristina Pop*
5. *app.smartdraw.com – for UML package diagram*
6. *app.creately.com – for the use case diagram*
7. [*www.stackoverflow.com*](http://www.stackoverflow.com)
8. [*www.geeksforgeeks.com*](http://www.geeksforgeeks.com)
9. *app.diagrams.net – for UML class diagram*
10. *Baeldung*