DOCUMENTATION

ASSIGNMENT *3*

STUDENT NAME: Cupsa Bogdan

GROUP: 30431

# CONTENTS

[1. Assignment Objective 3](#_Toc128043139)

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

[3. Design 3](#_Toc128043141)

[4. Implementation 3](#_Toc128043142)

[5. Results 3](#_Toc128043143)

[6. Conclusions 3](#_Toc128043144)

[7. Bibliography 3](#_Toc128043145)

# Assignment Objective

Design and implement an application for managing client orders for a warehouse.

The secondary objectives are:

* Analyze the problem and identify requirements.
* Design the orders management application.
* Implement the orders management application.
* Test the orders management application.

# Problem Analysis, Modeling, Scenarios, Use Cases

For this project, the functional requirements are:

* The application should allow users to select a client, product, or orders page from the menu.
* The application should allow users to add, edit or delete a client.
* The application should allow users to add, edit or delete a product.
* The simulation should allow users to add orders.

Use cases:

A picture containing diagram, sketch, drawing, line

Description automatically generated

Actor: User.

Use Case: Add product.

Main Scenario:

1. Enter the name in the name field.
2. Enter the price in the price field.
3. Press add button.

Use Case: Edit product.

Main Scenario:

1. Enter the name in the name field.
2. Enter the price in the price field.
3. Select the product to be updated.
4. Press edit button.

Use Case: Delete product.

Main Scenario:

1. Select the product.
2. Press delete.

Use Case: Delete client.

Main Scenario:

1. Select the client.
2. Press delete.

Use Case: Add client.

Main Scenario:

1. Enter the name in the name field.
2. Enter the address in the address field.
3. Press add button.

Use Case: Edit product.

Main Scenario:

1. Enter the name in the name field.
2. Enter the address in the address field.
3. Select the product to be updated.
4. Press edit button.

Use Case: Make order.

Main Scenario:

1. Select Client.
2. Select Product.
3. Enter the quantity in the quantity field.
4. Press on create order.

Use Case: Make order.

Secondary Scenario:

1. Select Client.
2. Select Product.
3. Enter bigger quantity.
4. Press on create order.
5. Insufficient stock error displayed.

# Design

The project is organized in packages: model, dao, bll, view, connection. Connection is responsible for connecting to the database (Sqlite). In the model we have the actual data. The dao package gets the data from the database. The bll package is used to abstract the database operations in case the database changes. The view model is responsible for the user interface.

Relations:

* Dependency between ProductCRUDView and TableUtils.
* Dependency between OrderCRUDView and TableUtils.
* Dependency between ClientCRUDView and TableUtils.
* Composition between ProductCRUDView and MainView.
* Composition between OrderCRUDView and MainView.
* Composition between ClientCRUDView and MainView.
* Association between ClientCRUDView and ClientBLL.
* Association between ProductCRUDView and ProductBLL.
* Association between OrderCRUDView and OrderBLL.
* Dependency between ProductDAO and ProductBLL.
* Dependency between ClientDAO and ClientBLL.
* Dependency between OrderDAO and OrderBLL.
* Association between ProductDAO and Product.
* Association between OrderDAO and Order.
* Association between ClientDAO and Client.
* Dependency between ProductDAO and ConnectionFactory.
* Dependency between OrderDAO and ConnectionFactory.
* Dependency between ClientDAO and ConnectionFactory.

A picture containing text, diagram, plan, technical drawing

Description automatically generated

Data structures used:

* List
* ArrayList
* Array

# Implementation

For the implementation of the polynomial calculator, I have the following packages:

* model
* dao
* bll
* view
* connection

connection:

ConnectionFactory is responsible for connecting to the database. The database I am using is sqlite, so for this connection, we don’t have to provide a username and password.

model:

In the model package I have the following classes:

* Client: id, name address and phone.
* Product: id, name, description, price, and stock,
* Order: id, client, product, quantity and date.

dao:

Here we find the classes responsible for working with the database, performing queries and getting the data.

* ClientDAO.
* ProductDAO.
* OrderDAO.

bll:

This is the abstract layer that should exist in case the database changes in the future. The following classes implement the methods responsible for abstractization of the DAO layer (not performing queries directly).

* ClientBLL
* ProductBLL.
* OrderBLL.

view:

The layer responsible for the user interface:

* MainView: in this page, the user selects where he/she wants to make modifications.
* TableUtils: The class is responsible for the reflection techniques. A table template is created that will then be used in all client, product and orders user interfaces.
* ClientCRUDView: responsible for the user interface of the client.
* ProductCRUDView: responsible for the user interface of the product.
* OrderCRUDView: responsible for the user interface of orders.

# Results

In this section I will show the correctness of the implementation by providing some results.

A picture containing text, screenshot, font, line

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Conclusions

In conclusion, after implementing the polynomial calculator, I have learned or deepened the following:

* A better understanding of OOP principles.
* Understanding of working with databases.
* Connecting to a database, Sqlite in my case.
* Generating Javadoc.
* Using reflection.
* Code organization in packages.
* Better at designing the graphical user interface.

# Bibliography

<https://www.wiley.com/en-us/Pattern+Oriented+Software+Architecture%2C+Volume+1%2C+A+System+of+Patterns-p-9780471958697>

<https://www.baeldung.com/parameterized-tests-junit-5>

<https://docs.oracle.com/javase/tutorial/uiswing/>

<https://www.baeldung.com/java-arraylist>

<https://www.baeldung.com/java-string-operations>

<https://dsrl.eu/>