Documentation

Assignment 3

Cot Mihaita Marian

Group: 30421

1. Assignment objective
2. Problem analysing, Modeling
3. Design
4. Implementation
5. Results
6. Conclusions
7. Bibliography

Assignment Objective

The objective of this assignment is to create a working java application which manages the working system of a warehouse/business. This application should contain a way to create and store, update, delete and show the clients, products and orders placed.

The software development process consists of understanding the requirements which is part of the initial sequence.

Constructing the software is the second section of this approach and consist of: the design, which is related to thinking and choosing appropriate strategies for implementing all the requirements; the unit-coding and testing which refers to the actual implementation of the code, the coding style applied which is chosen as an oop approach.

Integration refers to constructing the functionality of the system by the usage of all the components created, this approach thinks of every component being separate and used for the main structure. Last but not least it refers to testing the entire system and putting it to work-

The following represent the sub-objectives:

A picture containing text, screenshot, font, number

Description automatically generated

A close-up of a document

Description automatically generated with low confidence

The main gimmick of this assignment was managing the data base using a gui, implementation which will be exemplified down bellow.

# Problem Analysis, Modeling, Scenarios, Use Cases

Chart of the system:

A picture containing text, diagram, line, plan

Description automatically generated

The chart from above shows how the problem works.

At first it describes how every class/package interacts with each other, similar classes such as Product/Client/Order have been placed in the same block for readability and simplification of the system.

The program start with the main view which is influenced by the main view controller, letting the user select between tinkering with the Clients/Products or Orders. The controller then sets up the next view based on the button clicked. The new frame which pops up is then influenced by the specific controller which sets up the final view on screen, containing the operation buttons as well as a JTable with all the data.

The functionality behind the frames go through a meticulous process, the data is verified through validators which are implemented using an abstract class and overridden methods for both Products and Clients. This classes require information from their respective models.

The basic operations are managed inside the DAO package which contains an abstract class, general and used by the other three classes. The operations need to directly interact with the database, setting up and closing respective connections.

Last but not least the models include also the table models which are created using an abstract model, customized later for what needs to be done.

# Design

A screenshot of a computer

Description automatically generated with medium confidenceOn the right side of the screen I put a picture of

the organization of the program which is sectioned as

it follows: busniessLogic, connection, dataAccess,

Model and presentation.

The whole project is based on an object oriented

Programming strategy, every class performing a

different action: the classes inside the businessLogic

package handle the validation of the data introduced

by the user. The connection class handles the methods

used to interact with the database such as opening and

closing certain connections.

The dataAccess package holds the classes which perform

the actual actions such as insertion, deletion, updating

and finding by certain id.

The model package consists of the structure of a Client/

Product/Order object which basically refers to a row

from a specific table. The Jtable models are created here

as well due to them being considered models used by the view

The presentation package holds all classes implicated in the creation of the gui, controllers and specified views.

A picture containing text, screenshot, font, number

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated with low confidence

With all due respect please ignore the garbage data inside the tables, this were used only for very early testing and will be removed later on.

Implementation

App: main class with the role of starting the actual program.

Model classes: Client with the following specifiers: id, name, address, email, age

Product: id, name, stock

Order: id, ClientID, ProductID, Quantity.

Each and every single one of them has specific getters and setters selected, they are superclasses and have at least two setting methods, one including the id and the other one without including a certain id.

Table models: created after an abstract class which gets all the data from inside a table and dumps it into a jTable.

DataAccess classes:

ClientDAO implements the following operations: insert into table, update a row, delete a row, and find client by id.

ProductDAO implements the following operations: insert into table, update a row, delete a row, and find product by id.

OrderDAO implements the following operations: insert into table, update a row, delete a row, and find order by id.

AbstractDAO: template for all the operations.

The queries are locally declared. Could implement them dynamically using reflexion but had no time for this right now sorry :3

Presentation classes:

View: main view of the system

ClientManagement: client frame, containing the fields, buttons and table specifically.

ProductManagement: client frame, containing the fields, buttons and table specifically.

OrderManagement: client frame, containing the fields, buttons and table specifically.

# Results

The implementation was tested using the garbage data showed above, the update of the order method was the trickiest one of them all since it had to temper with both the ex-order and the new one.

# Conclusions

This assignment was a bliss, a bit annoying because we had to use java for it but working with databases is always a pleasure. Surprisingly enough connecting to the database and creating the methods to temper with it was very simple, due to working on a maven project all I had to do was include a certain dependency, feature which deeply impressed me.

# Bibliography

1.Wiki polynomial long division: <https://en.wikipedia.org/wiki/Polynomial_long_division>

2. Geeks for Geeks, for tips on maps:

<https://www.geeksforgeeks.org>

3. Stack Overflow, place you can get answer to any problem:

<https://stackoverflow.com>