Domain model with comments -->***Adrian and Monika***

Transformation of the domain model to a relational model with comments -->***Adrian and Monika***

Fully dressed use-case description for order processing and relevant documentation for that use-case. --> ***Alex and Kristupas***

Interaction diagrams for the fully dressed use-case -->***Monika***

Mock-ups, argumentation for the design

SQL scripts for creation of the database.

SQL scripts for insertion of data into the database.

Code standard

The testclasses has to be part of the Eclipse project.

~~Based on the domain model a relational model is to be designed (mapping/transformation).~~

Script(s) for creating the database and insertion of data are to be written.

Java programs are to be created so it is possible to carry out CRUD operations on the Customer and Product classes.

Finally the use case for order processing is to be described, designed, implemented. Beside that you have to make test classes.

*Domain Model*

The first thing we decided to edit in the domain model was to add a saleLine between the product and the SalesOrder which would contain the amount of items purchased of that type and the price of a single unit.

Then while thinking about the database we were going to connect our system to we decided that there should be another class called address. It would contain zip cod, city and the street. This class then would be connected to customers, orders for deliveries and suppliers.

The SalesOrder at first was supposed to always have an invoice. However, after thinking more about it we found out we first create an order and only then an invoice. This means the multiplicity in the domain model had to be changed from 1 to 0..1 .

*Relational Model*

The relational model shows us exactly how we are going to connect our tables in our database. We have started from the domain model by discussing how we should design the database.

One of the things discussed was about the way we should connect the customer to the order. We have decided to connect our “customer” table directly to the “order” table. The other proposal was to connect the customer to the “invoice” table and every time we have an order we assign it to an invoice. Our decision is based on the most efficient way of accessing the orders of a customer.

Also, we have normalisated our database by creating the “address” table.

The relation “many to many” between product and supplier was solved by creating an additional table where we connect these two tables.