Mock-up

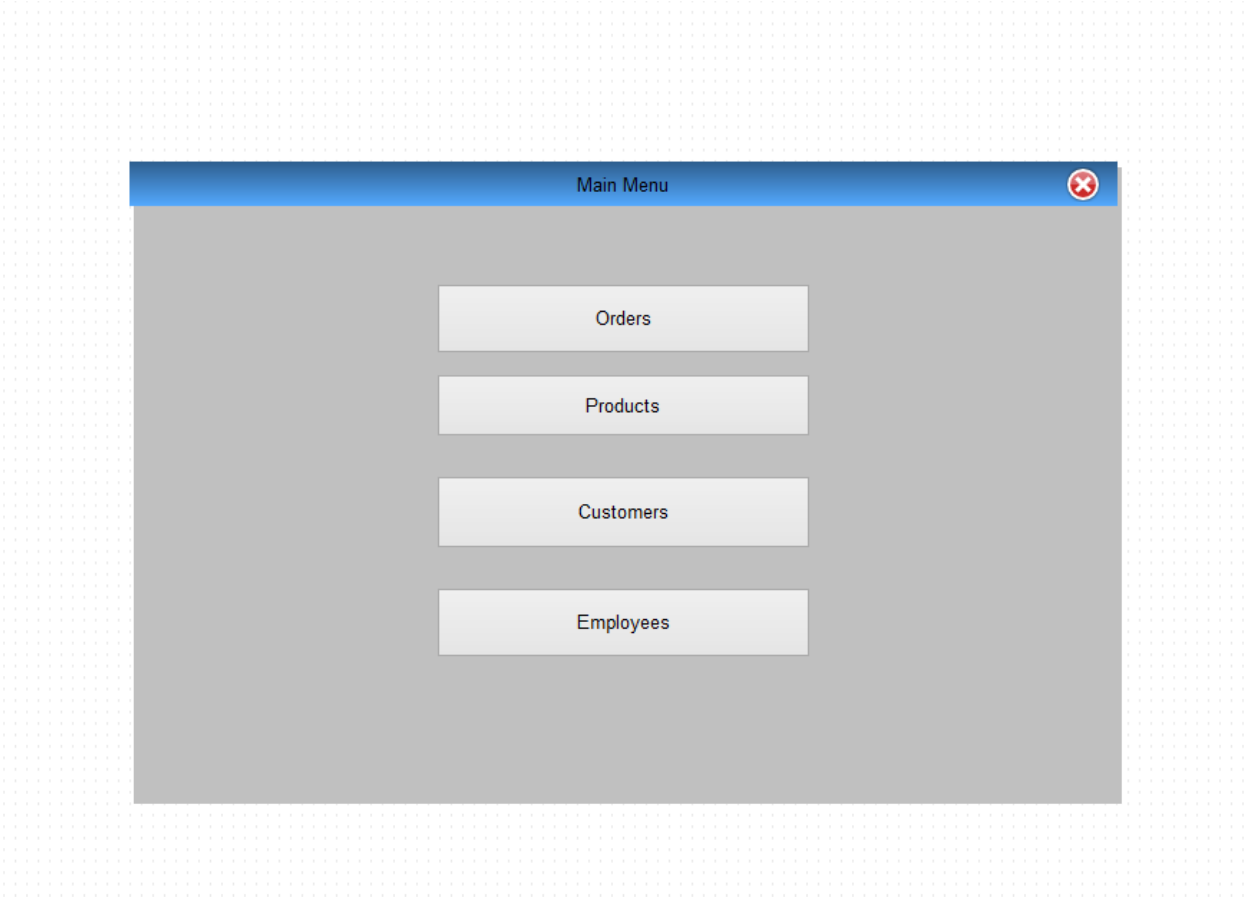


fig. 1 - Main Menu mock-up

As time progressed, it was decided that there'd be too much code behind the "Products" menu, so it was split into its respective sub-classes. The "Employees" menu was removed as it was not the main focus of this project. "Orders" currently exists in the program as "Sales". "Suppliers" was also added.

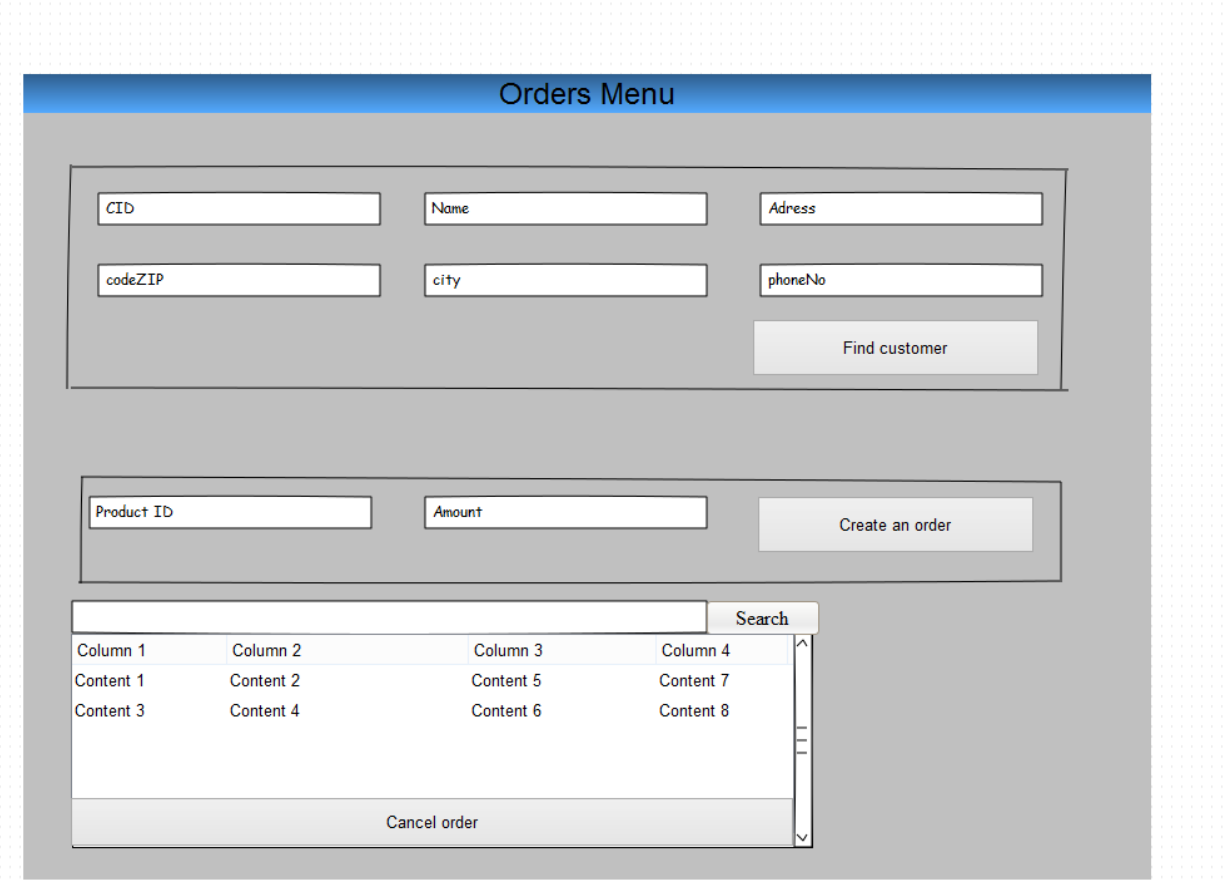


fig. 2 – Orders Menu mock-up

Fully dressed

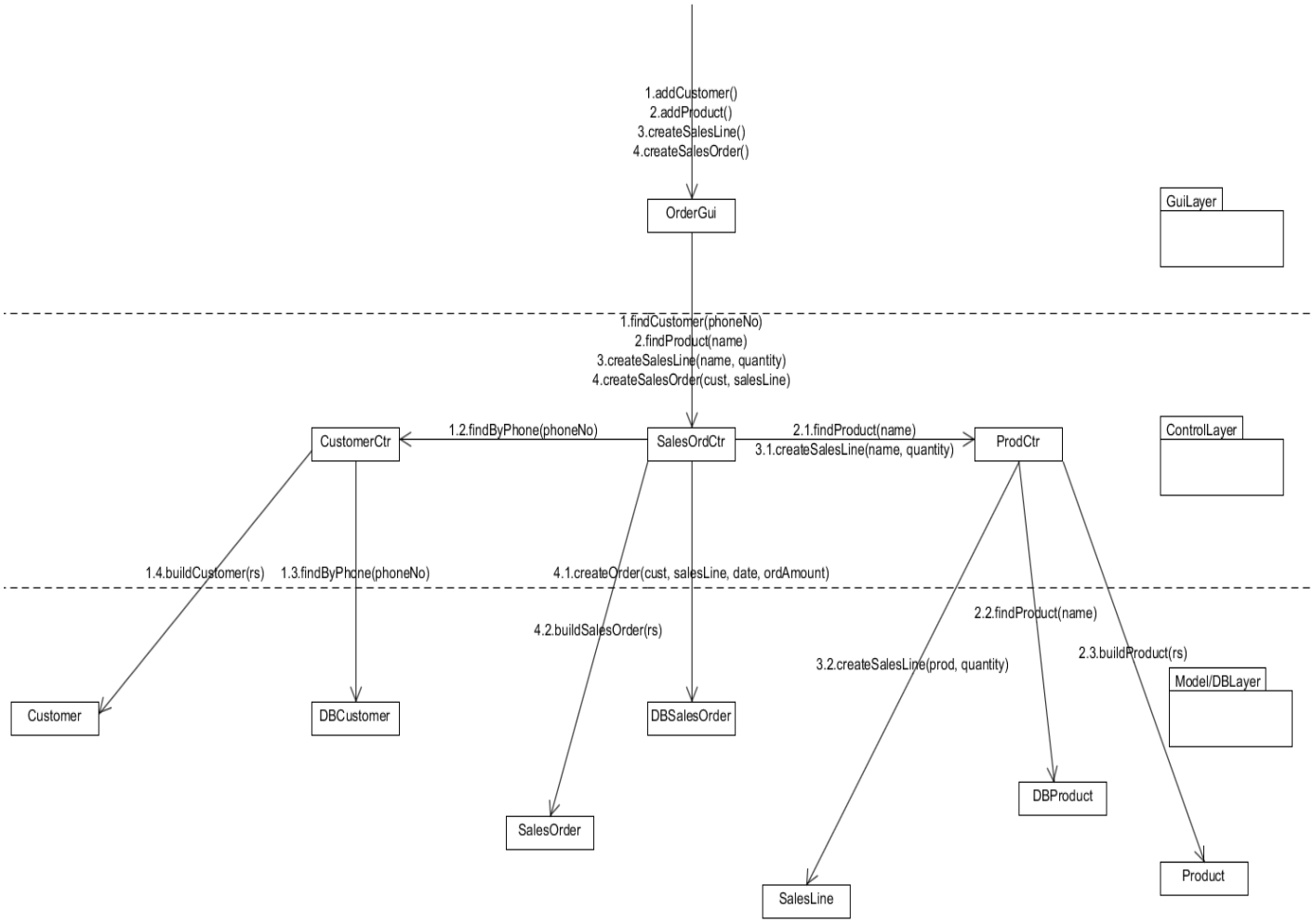
Use case	Create order
Actor	Employee
Pre-conditions	The product already exists.
Post-conditions	The order has been made and the product(s) has been assigned through the sales' line.
Flow of events	**System**
1. The customer either calls or e-mails to buy. 2. The employee creates the customer 3. The employee makes an order and adds the customer's ID to it. 6. Each and every product is packaged in a different box.	4. The products and their amount are added to the "salesLine" 5. They're added to the order, along with the ID of the customer. 7. The delivery status changes to "pending".
Alternative flow	2a. The product(s) is not available for the date wanted, so the employee contacts the customer for further information

The employee is "acting" without the Customer, because he/she's the one that can input information in the program. The pre-conditions are that only a product exists, because it's very likely for the customers to not be registered before buying. Step 3 is divided into Step 4 and 5 in the system due to the employee not seeing how the events unfold. For him/her the "salesLine" is not obvious, there is just an order.

Step 7 does not complete the order in real life and that is due to the coffee break principle with the use cases.

2a happens due to the possibility of the communication being via e-mail.

Interaction Diagram



Before everything used to connect in the model layer (ex. Customer and Customer Container), now it goes to the DB Server, back to the controller and then to the class itself in the model layer.

Domain Model

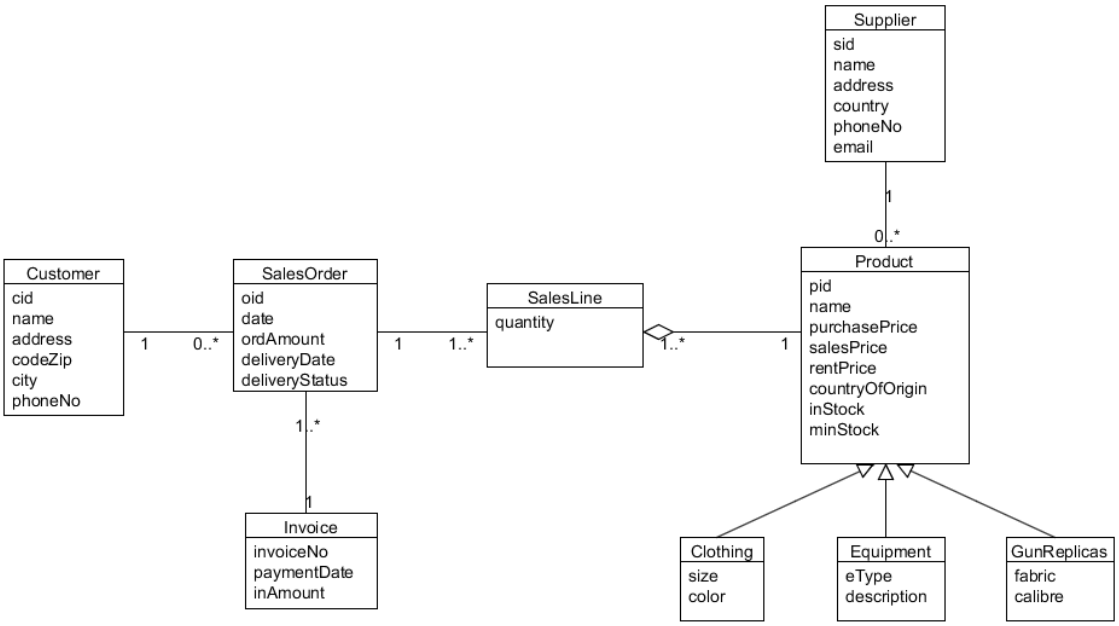
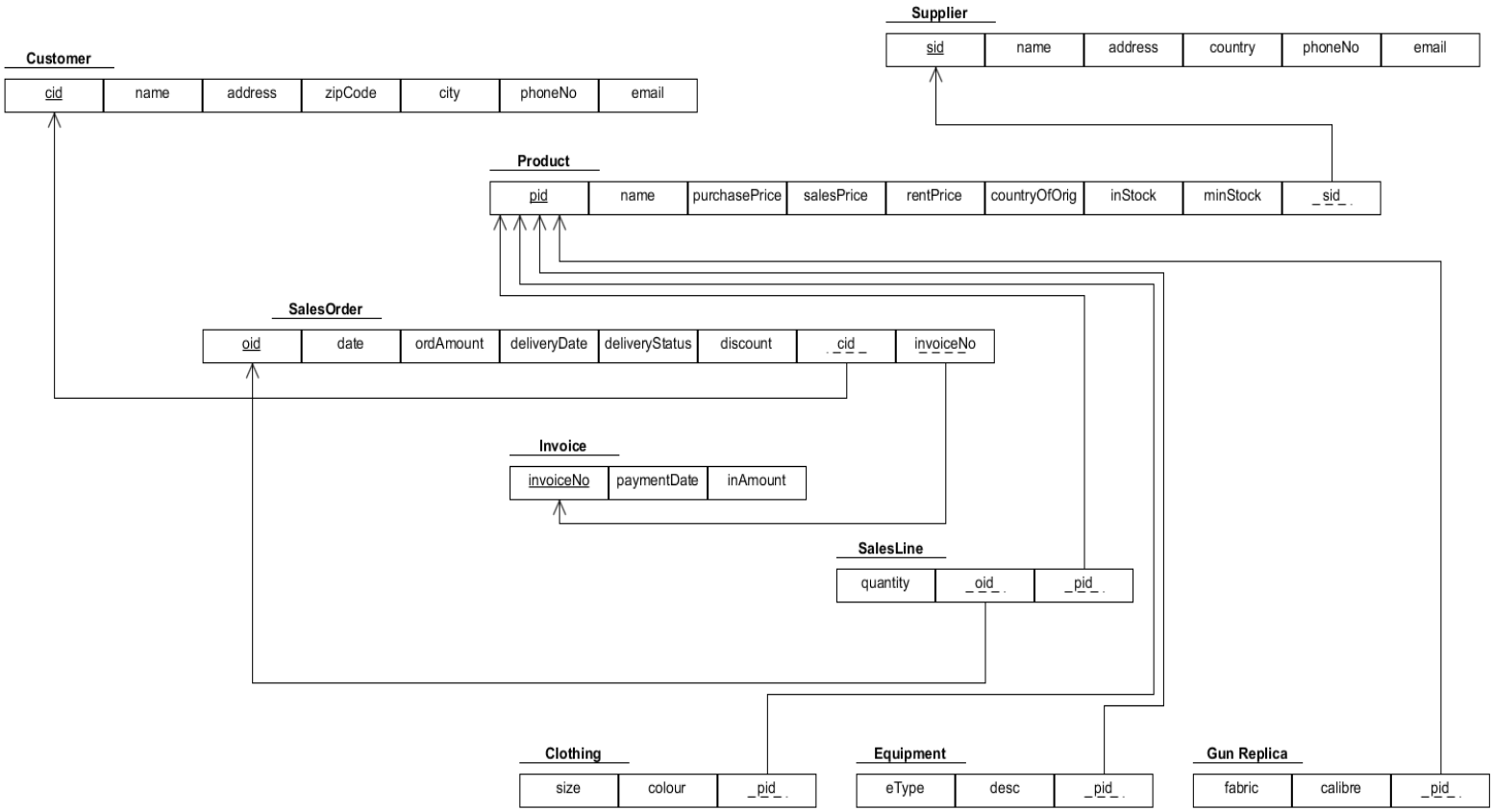


fig. 2 - Domain model

Dummy ID's, "SalesLine" and "inStock" were added. "SalesLines", because we needed a different amount for each product in the "SalesOrder" – not 5 of each and every one. inStock (variable in "Product"), because the company needs to check all the available units they have, when they're about to sell more than 2 units of a product.

Relation model



Taking in mind the multiplicities, we added foreign keys on the side with the one to many(*) sign. We also added dummy ID's.

Repository: https://github.com/IliyanStoev/dmai0914_2Sem_2

Remarks: We didn't do the Test Classes and we only have the insertion method for Customers. The program lacks some functions and we'll try to fix them until the evaluation comes.