Object Relational Mapping with JPA

## About this exercise

*This exercise was originally designed as a preparation for a two-hour exam, so it does not represent the complexity of the new 24 hours exam. Expect to spend 2-4 hours since right now, you are learning, so complete it Friday. Also, and important, you are not all expected to complete it from A-Z. It was meant as an example of a real exam exercise and they have to cover a span from the grades 2-12.*

|  |  |
| --- | --- |
|  | [*Watch this before you start*](https://cphbusiness.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=51d20fb5-7520-40a7-a219-aacc013a4b69) |
|  |  |

## *All answers are in Times New Roman red text. Like this.*

## General part

* *Explain the rationale behind the topic Object Relational Mapping and the Pros and Cons in using ORM*

*It’s supposed to be simple. I find it annoying, because of its vulnerability in Netbeans. Broke a the program by making a string field into a integer field. Even after changing all the past references, it still threw compatibility errors. Swapping the field location of the integer field and another String field solved the issue for some reason, even though it was out of the table relations context.*

*Updating and creating is also merged into one. Which makes it harder to control and more annoying to fix errors.*

*Explain the JPA strategy for handling Object Relational Mapping and important classes/annotations involved.*

*The entitymanagers, persistence and queries; bi-directional and uni-directional relationship annotations. Using JPA queries with ORM seems like an ideal choice.*

* *Outline some of the fundamental differences in Database handling using plain JDBC versus JPA*

*JPA seems more automated. Having to think about statements being insecure isn’t an issue in JPA, due to the structure of the queries. The syntax was different.*

## Practical part

This model represents an initial model for a system that can handle orders. Order refers to a customer and a number of order lines. Each order line has a quantity and it refers to an item type. The item type has a name, a description and a price. The price for each order line is the quantity times the price. The total price for an order is the sum of all its order lines.

1. Examine and understand the diagram.
2. Create a Maven Java Application with NetBeans, and use Object Relational Mapping (JPA) to implement the OO classes and the corresponding Database Tables[[1]](#footnote-1).
3. Create a façade and implement as many of the methods below as you have time for, not necessarily in the given order:

* Create a Customer
* Find a Customer
* Get all Customers
* Create an ItemType
* Find an ItemType
* Create an Order and Add it to a Customer
* Create an OrderLine for a specific ItemType, and add it to an Order
* Find all Orders, for a specific Customer
* Find the total price of an order ….

1. You don't necessarily need to implement all Entity-classes before you start on part-3, even if the overall problem doesn't make real sense without all. Make sure to implement some of the methods in part-3. [↑](#footnote-ref-1)