System Development 2

**Exercise: Towards Mini Project Persistence**

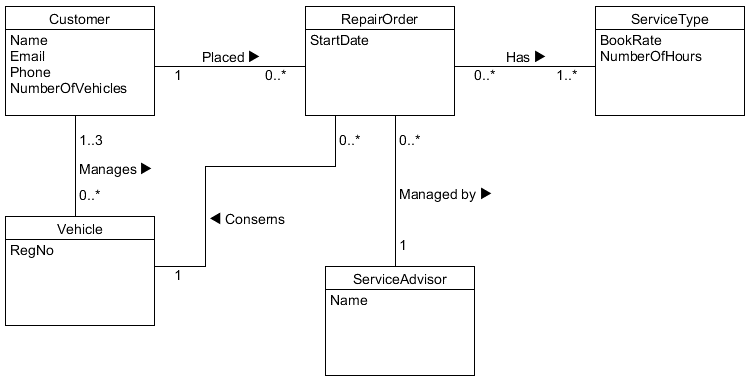
### 1. Domain model and relational model – Automobile repair shop

The exercise below is from a previous exercise

|  |
| --- |
| .. Domain model and relational model: automobile repair shop\* You are designing a software system for an automobile repair shop. When a customer brings in a vehicle, a service advisor will write up a repair order. This order will identify the customer and the vehicle, along with the date of service and the name of the advisor.  A vehicle might need several different types of service in a single visit. These could include oil change, lubrication, rotate tires, and so on. Each type of service is billed at a pre-determined number of hours work, regardless of the actual time spent by the technician. Each type of service also has a flat “book rate” of dollars-per-hour that is charged.  Draw a domain model and map it to a relational model.  \*based on:  http://www.tomjewett.com/dbdesign/dbdesign.php?page=repair.php  http://www.tomjewett.com/dbdesign/dbdesign.php?page=software.php |

**Domain model**

Based on the text this domain model was constructed:



Examine the domain model based on the text.

If you think the diagram is good you can use that in the further work.

Otherwise you might modify it based on the RepairShopDomainModel.uxf file

**Relational database schema**

Map the domain model to a relational database schema using the transformation rules.

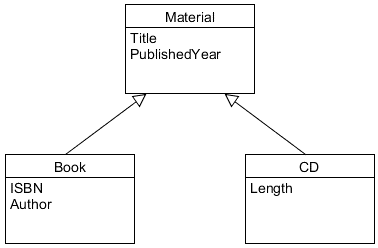
**Possibly implement the database**

Write sql-scripts corresponding to the relational database schema.

Implement the database and insert some data.

### 2. Domain model and relational model – Library

### We have this domain model:



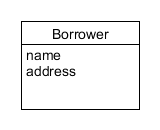
UMLet file: LibraryGeneralizationDomainModel.uxf

Transform the domain model to the relational model.

Possibly implement the relational model.

### 3. Interaction diagrams – Borrower

From the domain model for a library system, we have this class:



You must now design some of the database access based on the layered architecture learned in Programming.

You consider these classes:

* BorrowerUI
* BorrowerCtr
* BorrowerDB
* Borrower

*Inserting a Borrower*

**Please draw an interaction diagram** for inserting a Borrower into the Library database:

createBorrower(id, name, address)

**Then implement the code** precisely following the interaction diagram (full traceability is expected).

To save time use:

* LibraryAsInteractionDiagram\_src.zip (Java code to work on)
* CreateLibraryAsInteractionDiagram.zip (Sql-scripts to create the database incl. tables)

*Retrieving a Borrower*

When done **draw an interaction diagram** for retrieving a Borrower by id.

Then implement the code precisely following the interaction diagram (full traceability is expected.