Exercises: Unit Testing and Isolation

Problems for exercises and homework for the "Java Web Development Basics" course @ SoftUni. Submit your solutions on the **course page** of the **current instance**.

Car Dealer – Back-End

Car Dealer is an application for cars and car parts sales accounting. It manages customers, cars, parts, suppliers and sales. You have been given the infrastructure of the back-end – entities, repositories, services, configuration etc.

You will also be given a simple documentation of the application.

Documentation

1. Database Documentation

The database of the Car Dealer application supports 5 entities:

Customer

- Has a **`Id`** – a **UUID string**
- Has a **Name** – a **string**
- Has a **`Birth Date`** – a **`LocalDate`** object
- Has a **`Is Young Driver`** – a **boolean**

Car

- Has a **`Id`** – a **UUID string**
- Has a **Make** – a **string**
- Has a **`Model`** – a **string**
- Has a **`Travelled Distance`** – a **64-bit integer**
- Has **`Parts`** – a **collection** of **`Part`** objects

Supplier

- Has a **`Id`** – a **UUID string**
- Has a **Name** – a **string**
- Has a **`Is Importer`** – a **boolean**

Part

- Has a **`Id`** – a **UUID string**.
- Has a **Name** – a **string**
- Has a **`Price`** – a **`BigDecimal`** object
- Has a **`Supplier`** – a **`Supplier`** object.

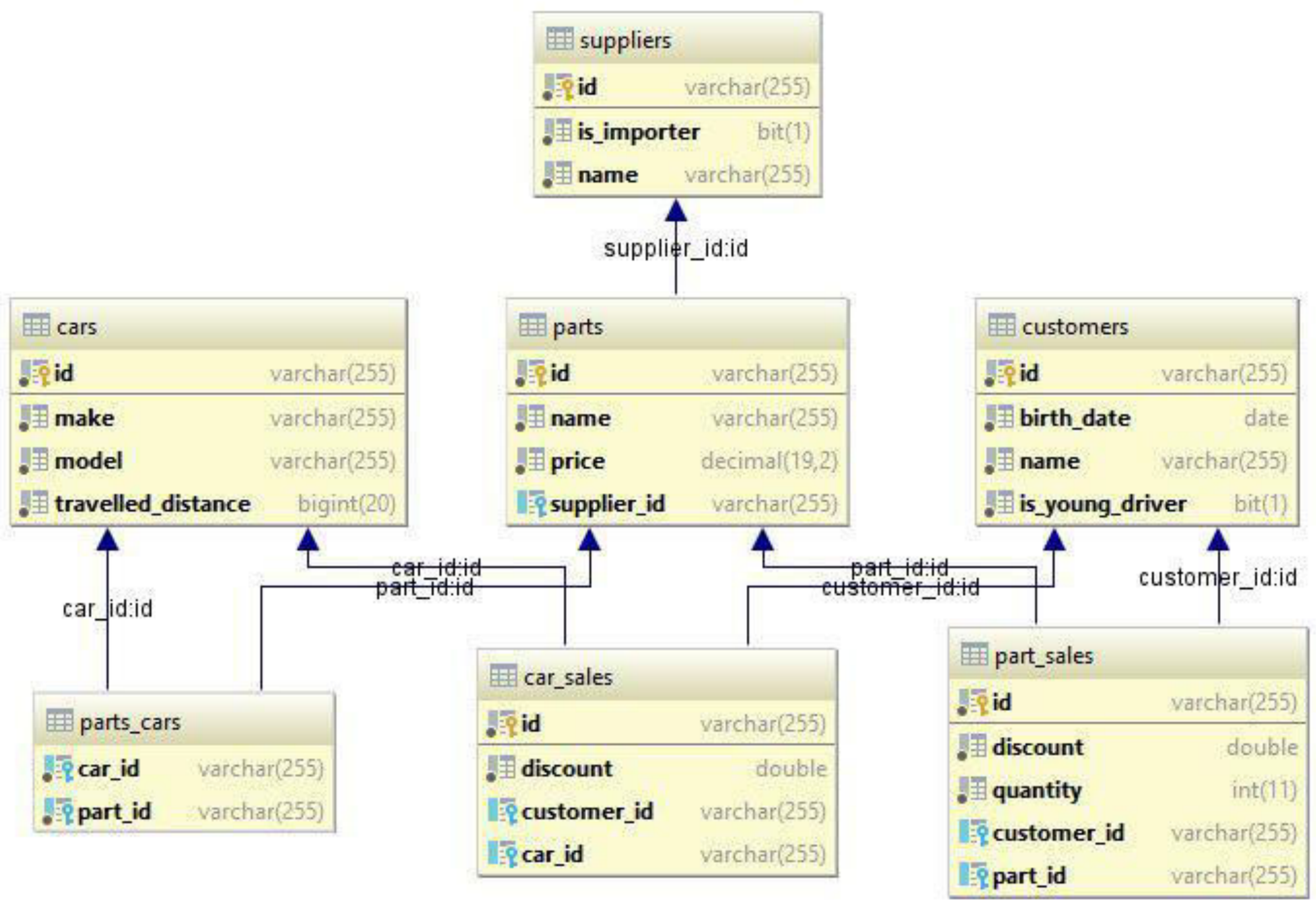
Car Sale

- Has a **`Id`** – a **UUID string**.
- Has a **`Discount`** – a **floating-point** (percentage)
- Has a **`Customer`** – a **`Customer`** object
- Has a **`Car`** – a **`Car`** object**.**

Car Sale

- Has a **`Id`** – a **UUID string**.
- Has a **`Discount`** – a **floating-point** (percentage)
- Has a **`Customer`** – a **`Customer`** object
- Has a **`Part`** – a **`Part`** object**.**
- Has a **`Quantity`** – a **32-bit integer**.

Here is a database schema:



2. Functionality Documentation

The functionality is self-explanatory if you look at the services. The database was more important. You will have to analyze the functionality by yourself to get familiar with the project. Most of the entities support casual CRUD operations, nothing more.

Service Testing

You have been tasked to perform Unit testing on the **Service Layer** of the **`Car Dealer`** application. Test all methods with all possible cases you can think of.

Use the **Code Coverage runner** of **`IntelliJ`** or a tool like **`EclEmma`** for **`Eclipse`** in order to calculate the coverage of your unit tests. Try to achieve a **Code Coverage** of **80%** on the **Services**.