Übung W3: JPA Assoziationen

Ausgangslage & Ziele
Diese Übung basiert auf demselben Projekt (lab-jpa.zip) in welchem Sie bereits im Unterricht mit dem Arbeitsblatt gearbeitet haben. Ziel dieser Übung ist es, das Problem der *referentiellen Integrität* zu verstehen.

Aufgabe 1)
In dieser Aufgabe gehen wir von einer (normalen) *bidirektionalen* 1:1 Assoziation zwischen den Entitäten `Customer` und `Address` aus und betrachten den folgenden Code:

```
Customer c1 = new Customer("Lee", 44);
Customer c2 = new Customer("Gosling", 55);
Address a = new Address("Dornacherstrasse", "Basel");
c1.setAddress(a);
c2.setAddress(a);

em.persist(a);
em.persist(c1);
em.persist(c2);
```

a) Ist der Aufruf `em.persist(a)` nötig? Falls ja, wie kann erreicht werden dass man diesen Aufruf weglassen kann?

b) Wie sehen die Einträge in der Tabelle CUSTOMER aus? Zeichnen Sie auch ein Objektdiagramm.

c) Erklären Sie, warum dieser Zustand nicht konsistent mit der Definition einer 1:1 Assoziation ist.

Aufgabe 2)
Das in der vorhergehenden Aufgabe diskutierte Problem, dass Assoziationen nicht konsistent sein können, tritt auch bei bidirektionalen 1:n Assoziationen auf. Wir definieren daher eine bidirektionale 1:n Assoziation zwischen den Klassen `Customer` und `Order`, d.h. einem Kunden sollen mehrere Bestellungen zugeordnet werden können.

```
Customer c1 = new Customer("Haller", 52);
Customer c2 = new Customer("Schneider", 66);

Order o1 = new Order();
Order o2 = new Order();
Order o3 = new Order();

c1.getOrders.add(o1); c1.getOrders.add(o2);
c2.getOrders.add(o1); c2.getOrders.add(o3);

em.persist(c1);
em.persist(c2);
```

a) Überlegen Sie sich, was Sie als Resultat erwarten, d.h. auf welche Bestellungen die beiden Customer `c1` und `c2` verweisen und vergleichen Sie ihre Erwartung mit dem effektiven Resultat.

Aufgabe 3)
Definieren Sie eine unidirektionale OneToMany Assoziation (z.B. zwischen `Order` und `OrderLine`) und erzeugen sie eine Bestellung (`Order`) mit 10 Bestellzeilen (`OrderLine`) Items. Laden Sie danach diese eine Bestellung und löschen Sie eine (z.B. die erste) Bestellzeile aus der Liste der referenzierten Elemente.

Geben Sie dabei die auf der Datenbank ausgeführten SQL Statements aus (setzen Sie dazu das Property `spring.jpa.show-sql=true`) und versuchen Sie das Resultat zu erklären. Wie sieht es aus, wenn Sie anstelle einer *Liste* ein *Set* verwenden?

Ist nach dem Entfernen eines `OrderLine`-Elementes aus der Liste der Bestellzeilen einer Bestellung dieses Objekt weiterhin in der Datenbank gespeichert? Und wie könnte es automatisch entfernt werden?

Aufgabe 4)
Neben diesen Aufgaben soll die Implementierung der JPA Schicht aus Übung W2 vervollständigt werden. Zur dieser Aufgabe noch folgende Hinweise:

a) Im Test `UserServiceTest.testDeleteUser` wird ein Benutzer gelöscht, was jedoch zu einer `DataIntegrityViolationException` führt. Der Grund ist, dass dieser Benutzer noch Ausleihen hat. Die Idee ist jedoch, dass beim Löschen eines Benutzers auch dessen Ausleihen gelöscht werden. Dies kann mit einer Kaskadierungsannotation erreicht werden.

b) Im Test `MovieRepositoryTest.testDelete` wird versucht ein Film zu löschen, aber dies führt zu folgendem Fehler:
`org.springframework.dao.InvalidDataAccessApiUsageException: Removing a detached instance ch.fhnw.edu.rental.model.Movie#5;`
Diese Ausnahme wird geworfen, wenn Sie eine *detached* Entität über den Entity-Manager löschen wollen. Die Entität muss vor dem Löschen in den Persistenzkontext eingefügt werden (z.B. mit der Methode `merge`).

Diese Änderungen sind trivial, aber nach diesen Änderungen sollten alle Tests ohne Probleme durchlaufen, und auch die MovieRental Client-Applikation sollte einwandfrei funktionieren (aber vielleicht finden Sie noch etwas, das ich übersehen habe). Damit haben Sie die Persistenzschicht erfolgreich mit JPA implementiert. Gratulation!

Ausgabe: 04.10.2021
Abgabe: 18.10.2021