JPA - mapowanie relacji

➔ **rodzaje relacji:**
- **One-to-One** (@OneToOne) - relacja jeden-do-jednego
- **Many-to-One** (@OneToMany, @ManyToOne) - relacja jeden-do-wielu
- **Many-to-Many** (@ManyToMany) - relacja wiele do wielu

➔ **kierunek relacji (dotyczy tylko Javy):**
- **Unidirectional** - jednokierunkowy
- **Bidirectional** - dwukierunkowy

➔ **mappedBy,** np.: @OneToOne(mappedBy = "address")
- jedna z klas w relacji **dwukierunkowej** jest **właścicielem** relacji
- mappedBy wskazuje na pole w klasie “po drugiej stronie” relacji
- dotyczy adnotacji: @OneToOne, @OneToMany, @ManyToMany
- JPA “wie” gdzie szukać danych do relacji.

➔ **cascade,** np.: @OneToOne(cascade = {CascadeType.ALL})
- zestaw operacji który ma być propagowany do klas zależnych
- ALL, PERSIST, MERGE, REMOVE, REFRESH, DETACH
- domyślnie nic nie jest propagowane

➔ **fetch,** np.: @OneToMany(mappedBy = "course", fetch = FetchType.EAGER)
- **LAZY** - lazily loaded, **EAGER** - eagerly fetched
- czy klasy w relacji mają być uzupełniane od razu czy później

https://www.thoughts-on-java.org/ultimate-guide-association-mappings-jpa-hibernate/

Zadanie nr 8:

1. Przejdź do modułu **jpa-starter** i sprawdź konfigurację bazy danych w pliku **persistence.xml** dla Persistence Unit o nazwie: **pl.sda.jpa.starter.relations.** Powinna wskazywać na bazę: **jpa_relations_test.** Stwórz nową bazę danych o takiej nazwie.

2. Przejdź do klasy ***JpaRelations*** i uruchom metodę *oneToOne().* Przeanalizuj kod. Następnie:
   a. odwróć przypisanie: *student.setAddress(address)* i zamiast niego przypisz do adresu studenta, sprawdź co się stanie
   b. usuń w klasie ***StudentEntity*** z adnotacji @OneToOne atrybut cascade, sprawdź co się stanie przy tworzeniu i zapisywaniu encji
   c. zmień nazwę kolumny z FK w tabeli ze studentami
   d. zmień właściciela relacji na klasę ***AddressEntity*** i pole *student*

3. Dodaj do ***StudentEntity*** nową relację typu **one-to-one** przenosząc do osobnej klasy (***SeatEntity***) informacje z pola *seat*. Dodaj odpowiednie adnotacje i sprawdź czy relacja działa. Nowa klasa powinna mieć trzy pola (poza id):
    a. *String columnNumber* - wartość kolumny, np.: 'A', 'B' …
    b. *int rowNumber* - numer rzędu, np.: 1, 2, 3
    c. *int seatNumber* - numer siedzenia, np.: 1, 2

4. Przejdź do klasy ***JpaRelations*** i uruchom metodę *oneToMany()*. Przeanalizuj kod. Następnie:
    a. zmień nazwę kolumny z FK (relacja z kursem) w tabeli ze studentami
    b. zamiast zapisywać studenta (*entityManager.persist(student))* zapisz w bazie tylko encję z kursem, sprawdź co się stanie
    c. stwórz nowy kurs, dodaj kilku studentów do kursu i zapisz zmiany w bazie. Zapisz tylko studentów i sprawdź czy kurs również został dodany.
    d. wyciągnij kurs, sprawdź czy ma studentów, usuń studenta o id: 1, zmień imię studenta o id:2 i dodaj nowego studenta do kursu. Sprawdź czy zmiany zapisały się w bazie.

5. Przejdź do klasy ***JpaRelations*** i uruchom metodę *manyToMany()*. Przeanalizuj kod. Następnie:
    a. spróbuj dodać parę nowych skilli i przypisz je do któregoś ze studentów
    b. usuń tego studenta i sprawdź czy usunięta została relacja między studentem i skillem
    c. **(dla chętnych)** ustaw relację student-skill jako dwukierunkową. Właścicielem relacji ma być obiekt SkillEntity
    d. **(dla chętnych)** wyciągnij jakiś skill z bazy i sprawdź jacy studenci są do niego dopisani.

6. **(dla chętnych)** Ustaw w PU parametr **hbm2ddl.auto** na 'none'. Stwórz w bazie danych: **jpa_relations_test** tabele:
    a. ***car*** (z polami: *id, name*) - jeden egzemplarza samochodu
    b. ***car_feature*** (z polami: *id, name*) - właściwości samochodu np. "kolor: czerwony", "rodzinny", "ekonomiczny", "z zacięciem sportowym" itp.
    c. ***producer*** (z polami: *id, name*) - dane producenta samochodu
    d. ***owner*** (z polami: *id, name*) - dane właściciela samochodu, załóżmy że właściciel może mieć tylko jeden egzemplarz samochodu

    Następnie dodaj do tabel relacje odpowiedniego typu:
    car - car_feature, car - producer, car - owner

    Stwórz encje na podstawie tabel podanych wyżej. Stwórz klasę **CarManager** i przetestuj swój kod zapisując dane w każdej z tabel.

7. **(dla chętnych)** Stwórz nową encję ***CoachEntity*** (z polami: *id, name*). Dodaj relację między trenerami a kursami. Relacja powinna być dwukierunkowa. Przetestuj swój kod:
    a. dodaj kilku trenerów do dwóch różnych kursów
    b. wyciągnij trenerów z kursu pierwszego i usuń relację między jednym z trenerów a konkretnym kursem
    c. wyciągnij wszystkich trenerów z danego kursu
    d. wyciągnij wszystkie kursy do których przypisany jest dany trener