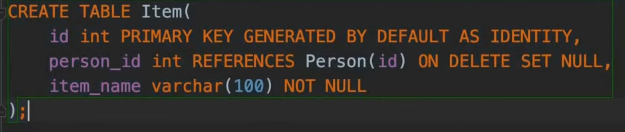
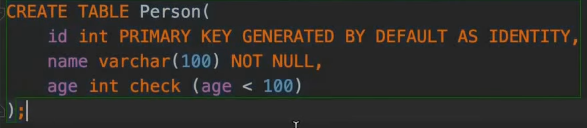
**Отношение One to Many в Hibernate**

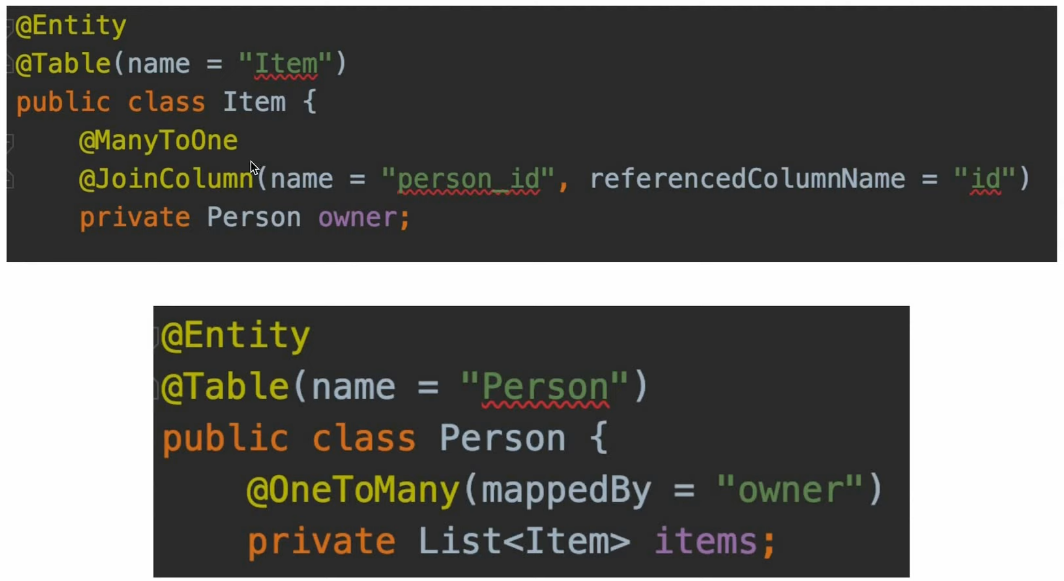
У нас есть связь один ко многим. У одного человека(**Person**) может быть множество товаров(**Item**).

**SQL команды для создания таблиц в БД:**





Чтобы выстраивать связи между сущностями в **hibernate** существуют специальные аннотации: **@ManyToOne, @OneToMany, @JoinColumn.**



**@OneToMany –** используется на родительской сущности.

Здесь мы указываем название того поля в классе **Item**, которое ссылается на нас. В классе **Item** поле которое устанавливает связь между товарами и людьми называется **owner -** это название мы подставляем в эту аннотацию в аргумент **mappedBy**.

**@ManyToOne** – используется на дочерней сущности.

**@JoinColumn** – обозначает колонку внешнего ключа. Так как внешний ключ у нас находится в таблице **Item** в нашей базе данных, поэтому аннотация **@JoinColumn** используется в классе **Item** в нашем **java** приложении. В этой аннотации мы должны указать в качестве арагументов:

**1. name** **-** название колонки внешнего ключа в нашей таблице, в нашем примере это **person\_id.**

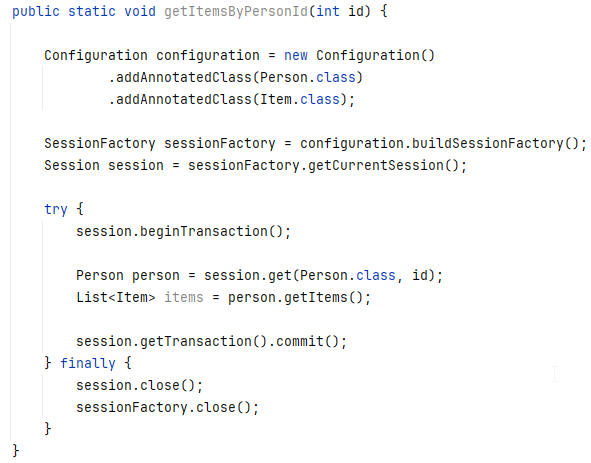
**2. referrencedColumnName –** название колонки в родительской таблице, тоесть на какую колонку ссылкается колонка с внешний ключом в дочерней таблице.

**Примечание:** иногда сторону которая владеет внешнем ключом называют **owning side**.

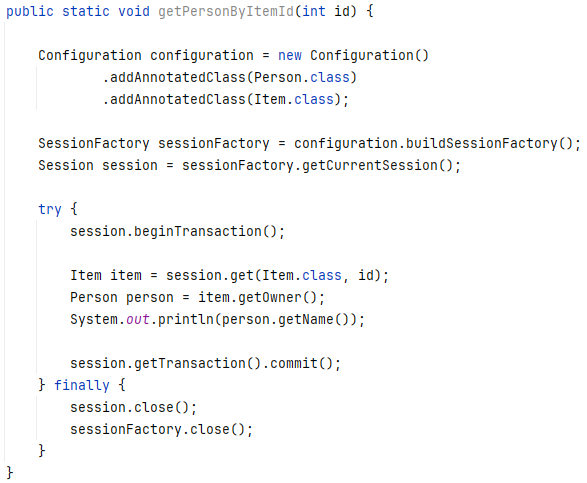
**owning side –** это та сущность в которой не используется аннотация с аргументом **mappedBy**.

**Примеры:**

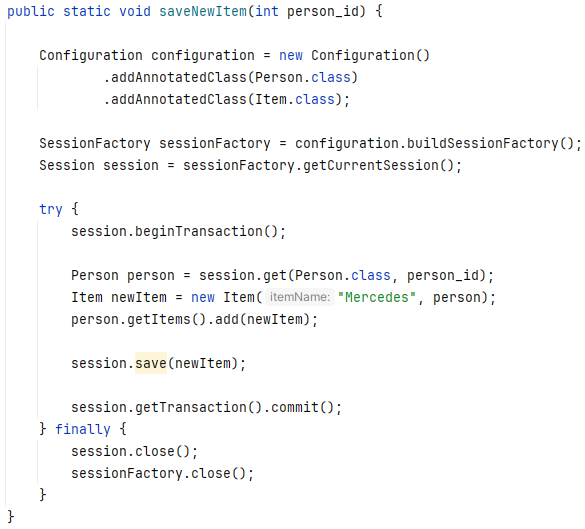
**1.** Получение товаров по id человека



**2.** Полуение человека по id товара



**3.** Создание нового товара и сохранение его в базу данных.



Порядок выполнения:

**1.** Получаем человека из базы данных (с помощью **hibernate**), этот человек будет владельцем товара, тоесть для него мы будем добавлять новый товар.

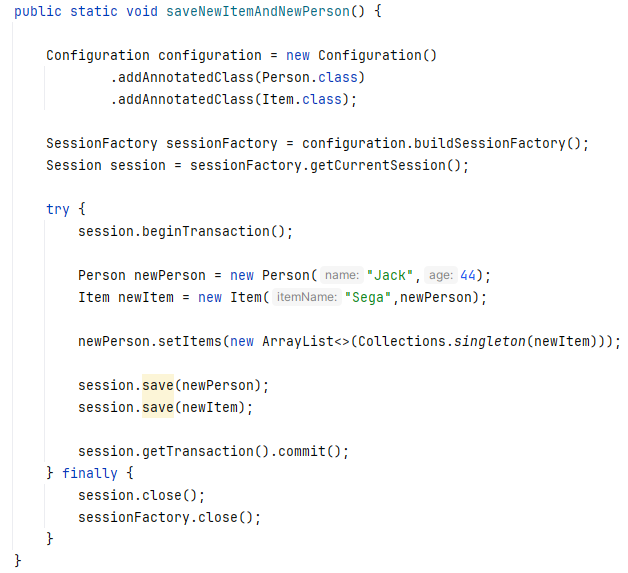
**2.** Создаем новый товар. Просто создаем новый объект класса Item. В качестве владельца укажем челвоека которого получили из базы данных.

Тот факт, что теперь у нас в этом объекте **item** в качестве владельца указан **person** – этого будет достаточно, чтобы в базу данных **hibernate** всё правильно записал. Тоесть он создаст новый **item** когда мы вызовем метод **save(newItem),** **hibernate** за нас создаст новую строку в таблице **Item** и в качестве значения внешнего ключа **person\_id** он укажет правильный **id** из объекта класса **person** которого мы указали при создании нового товара.

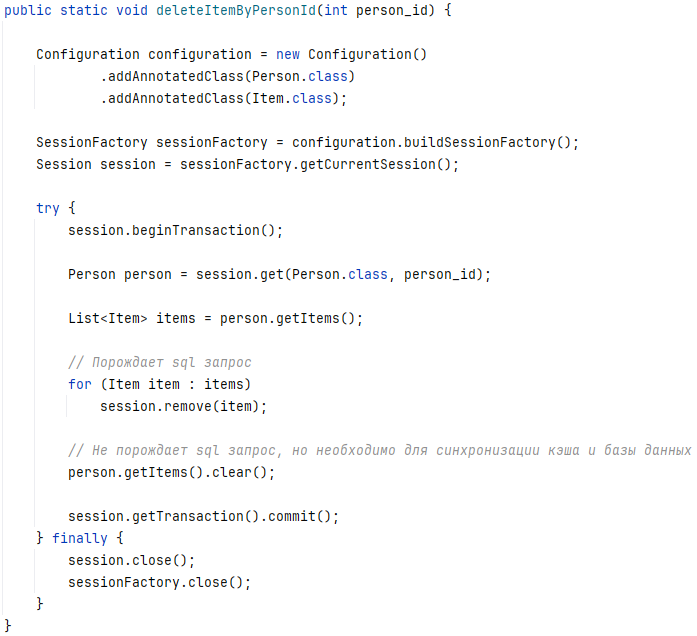
Тоесть мы оперируем строго с объектами, а **hiberante** за нас правильно выстраивает связи в наших таблицах базы данных.

**Здесь может возникнуть вопрос:** Мы указали, свзяь только на стороне товара, тоесть указали, что у товара есть человек, но так как у нас связь двухстроняя, не надо ли нам так же указывать связь на стороне человека ? Дело в том, что если нас интересует только то, что будет сохранено в базу данных, то код ***person.getItems().add(newItem);*** писать не надо потому-что **hibernate`a** заботит только то, что мы делаем на **owning side.**  У нас **item** – это **owning side** (сущность которая владеет внешним ключом) и поэтому **hibernate**`у не нужно, чтобы мы указывали связь между этими двумя сущностями с двух сторон, а достаточно указать связь только на стороне **owning side** с помощью кода ***Item newItem = new Item("Mercedes", person);***. **НО** здесь всё немного сложней. **Hibernate** кэширует объекты в своей памяти, тоесть каждый раз когда мы получаем какого-то человека, то **hibernate** не каждый раз делает запросы к базе данных потому-что он эти объекты кэширует в своей памяти, чтобы оптимизировать затраты по времени и если не установить связь так же на стороне человеке с помощью кода ***person.getItems().add(newItem);***, то возможно такое, что в таблице базы данных всё будет правильно(все данные запишутся), но если вы будете получать этого человека с **hibernate,** то будет выдаваться старая версию человека, тоесть та версия человека без добавленного товара, потому-что **hibernate** не будет делать запрос к таблице и не будет видеть, что у этого челвоека появился новый товар. Поэтому, чтобы значения в таблице и значения в кэше **hibernate`a** полностью друг другу соответствовали хорошей практикой считается задавать отношения с двух сторон, тоесть сначала задали отношение на стороне товара: ***Item newItem = new Item("Mercedes", person);***, и еще нужно задать отношение на стороне человека: ***person.getItems().add(newItem);*** - этот код никак не повляет на базу данных, эта строчка не пораждает никакие **sql** запросы, но эта строчка гарантирует, что наши объекты в кэше **hibernate`a** будут полностью соответствовать таблицам в базе данны.

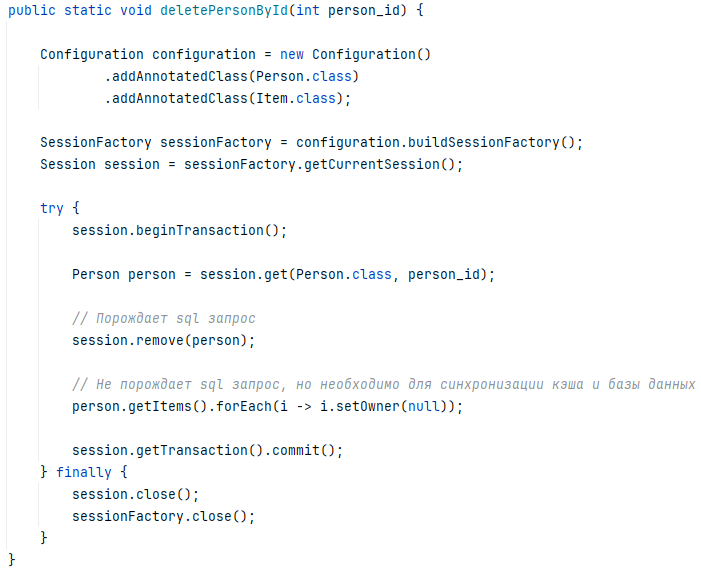
**4.** Создание нового товара и нового человека



**5.** Удаление товаров по **id** человека.



**6.** Удаление человека по его id.



**7.** Изменение владельца товара.

