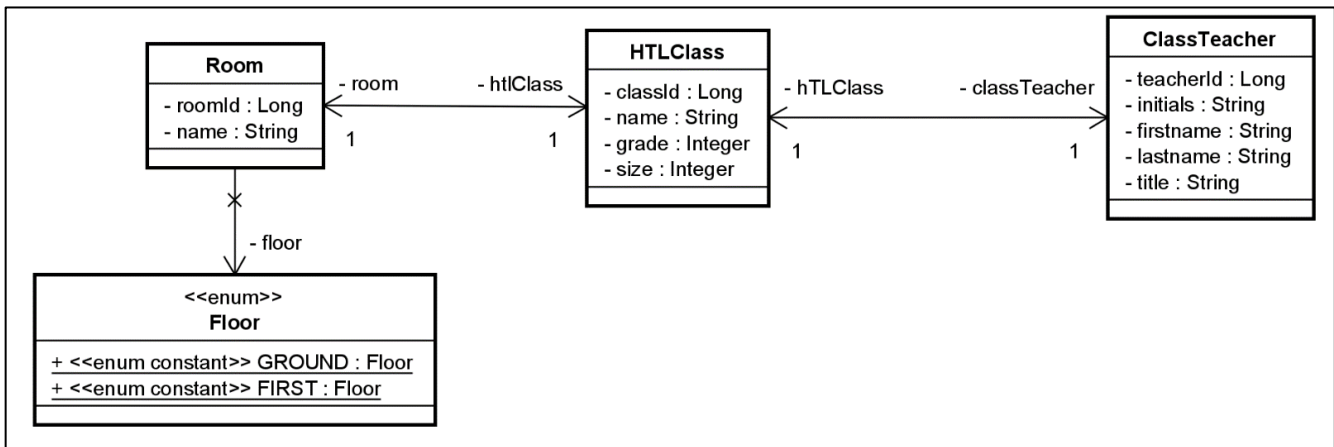


Example 102 JPA Classinfo

Implement a Spring-Boot-Data-JPA application for the postgres database **classdb**.
Create the Pojo classes according to the following CD:



Mapping constraints for the Pojo-classes:

- ❏ JPA-mapping according to the class diagram
- ❏ The primary-key value are created by a sequence in the database
- ❏ Apply appropriate cascade- and fetch-types to assoziations
- ❏ Always use the dot-operator for joins in JPA-queries when possible

pojos.Room.java

- ❏ Room name that starts with 1 are on the ground floor, room name that start with 2 are on the first floor

repositotries.RoomRepsoitory.java

- ❏ Create the following **Query**-Methods:
 - **findByClassName** to get a room specified by a classname
 - **findAll** to get all rooms
 - **findByFloor** to get all rooms of a floor (ground- or first-floor)
 - **countAll** to get the total number of rooms

pojos.HtlClass.java

- ❏ The enum-value for **floor** is stored as a varchar in the database

repositotries.HtlClassRepsoitory.java

- ❏ Create the following **Query**-Methods:
 - **findByName** to get a class specified by its name
 - **findAll** to get all classes
 - **findByFloor** to get all classes of a floor (ground- or first-floor)
 - **countAll** to get the total number of classes

pojos.ClassTeacher.java





repositotries.ClassTeacherRepsoitory.java

- ❏ Create the following **Query**-Methods:
 - **findByName** to get all teacher specified by a name pattern
 - **findByClassname** to get a teacher specified by the classname
 - **findByGrade** to get all teacher specified by a grade
 - **countAll** to get the total number of teachers

pojos.Floor.java

- ❏ Enumeration with the values **GROUND** and **FIRST**

init.DbManager.java

-  implements the **ApplicationRunner** Interface
-  The method **importTables()** is only called only when:
spring.jpa.hibernate.ddl-auto=create in file **application.properties**
-  **importTables()** - Read values from file **schooldata.json**. Distribute the values to the different Pojo-objects (use **JsonNode**) and persist them to the database **classdb**. Make sure that all bidirectional associations are setup! Use Repository-Queries to print successfull import to the console:
 rooms imported: 20
 classes imported: 20
 classteacher imported: 20
-  **testQueries()** test all query-methods from the repositories and print the results to console.