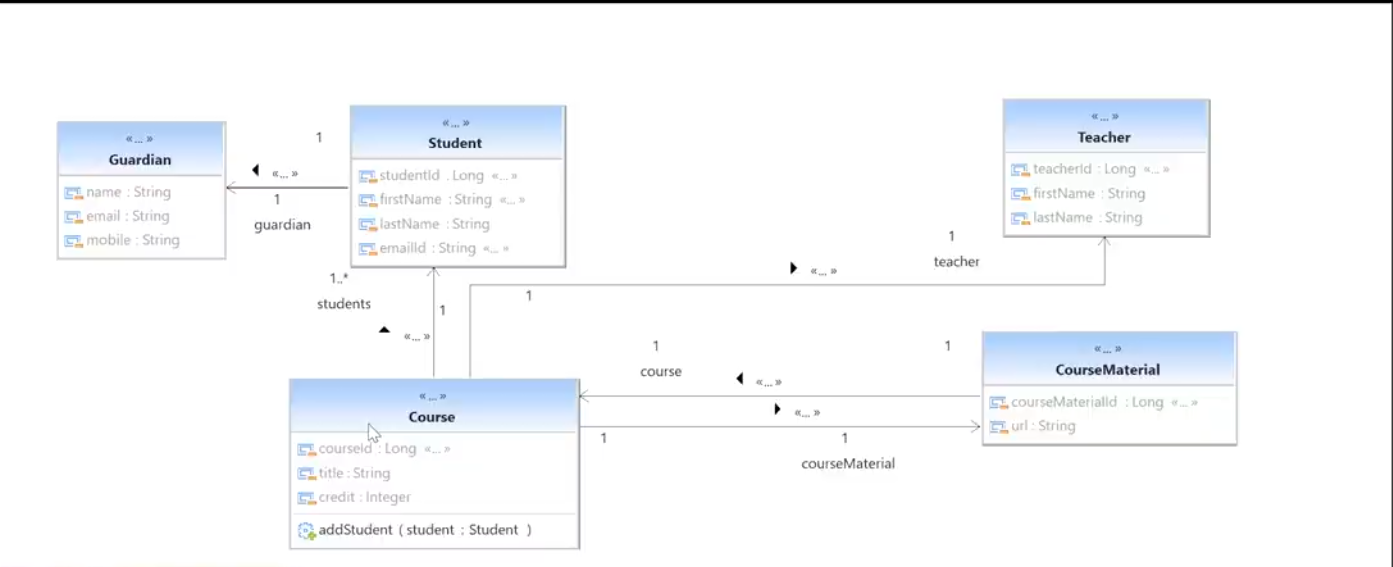
**ORM –** Object Relational Mapping i.e. mapping your objects to tables on the database.

You’re supposed to use Jpa using any of the third party ORM specifications for example Hibernate.



<https://start.spring.io/#!type=maven-project&language=java&platformVersion=3.2.1&packaging=jar&jvmVersion=17&groupId=com.emmakatwebaze&artifactId=spring-data-jpa-practice&name=spring-data-jpa-practice&description=Demo%20project%20for%20Spring%20Boot&packageName=com.emmakatwebaze.spring.data.jpa-practice&dependencies=data-jpa,lombok,sqlserver,web>

It’s not advisable to use ;

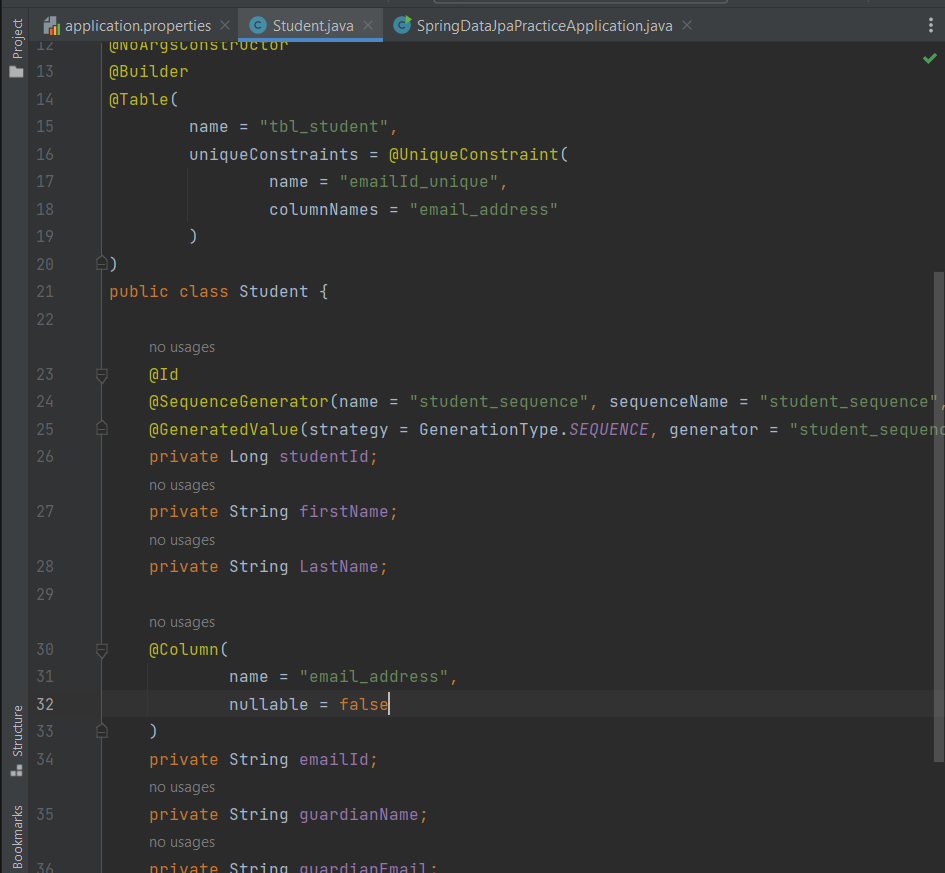
spring.jpa.hibernate.ddl-auto=update

in a production application.

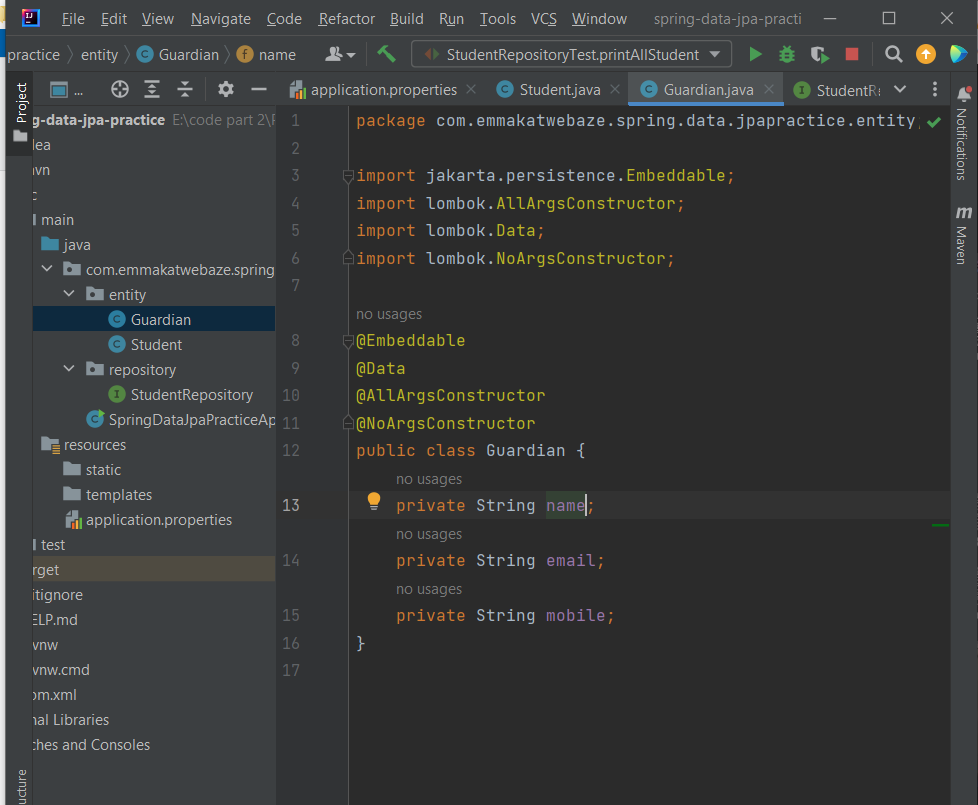
It’s advised in a production grade application to state the name of your table as well as the name of each column with @Column annotation. You should also use a Sequence annotation to Generate your unique ids.



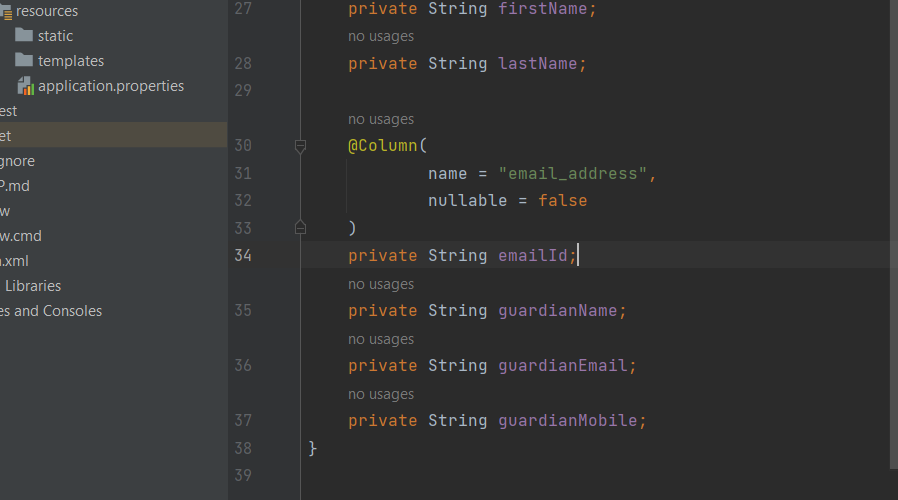
You can add the unique constraints on email as well as make sure it’s not null by:



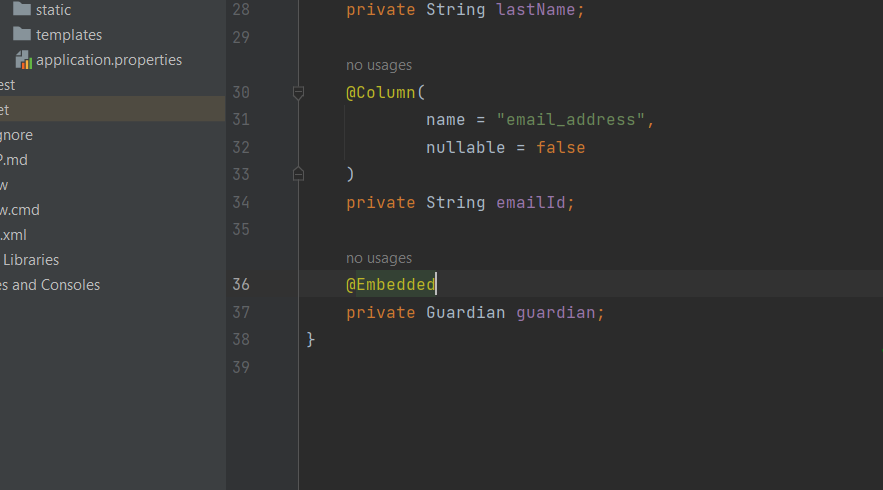
You need a new class for the guardian properties hence you’re supposed to create an @Embeddable class as below;

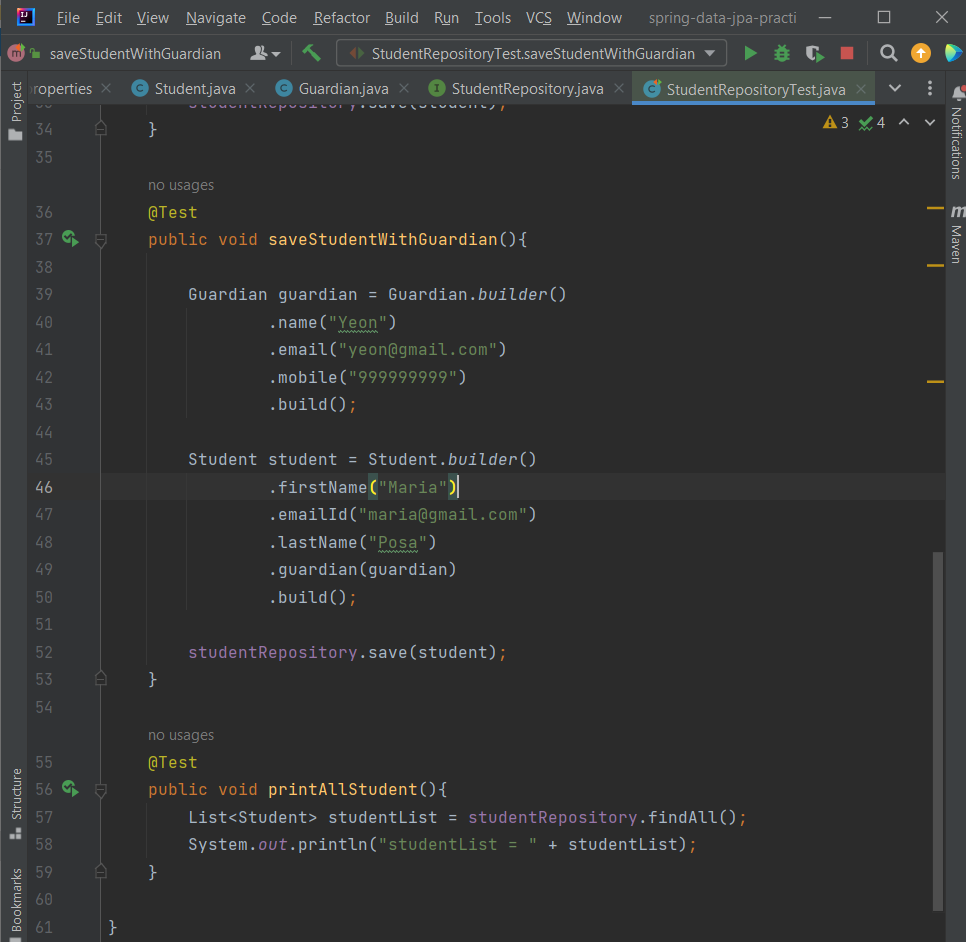
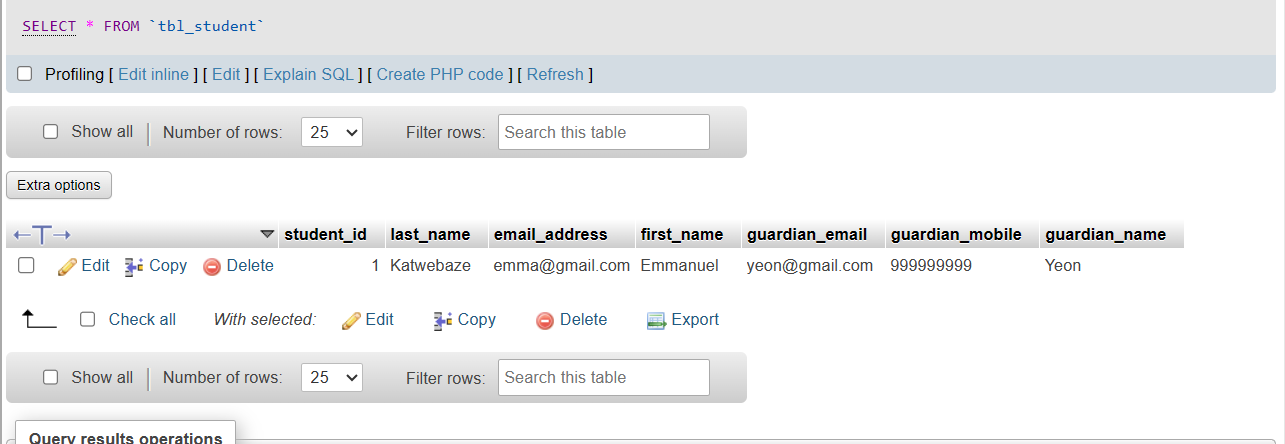
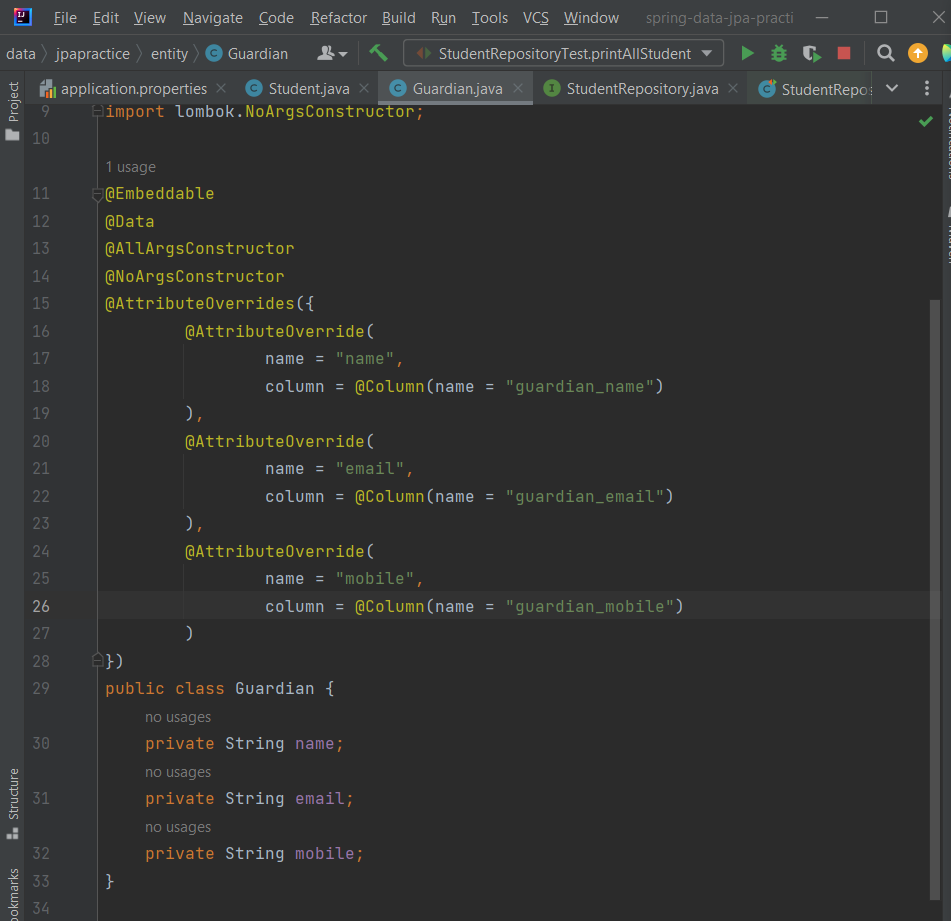


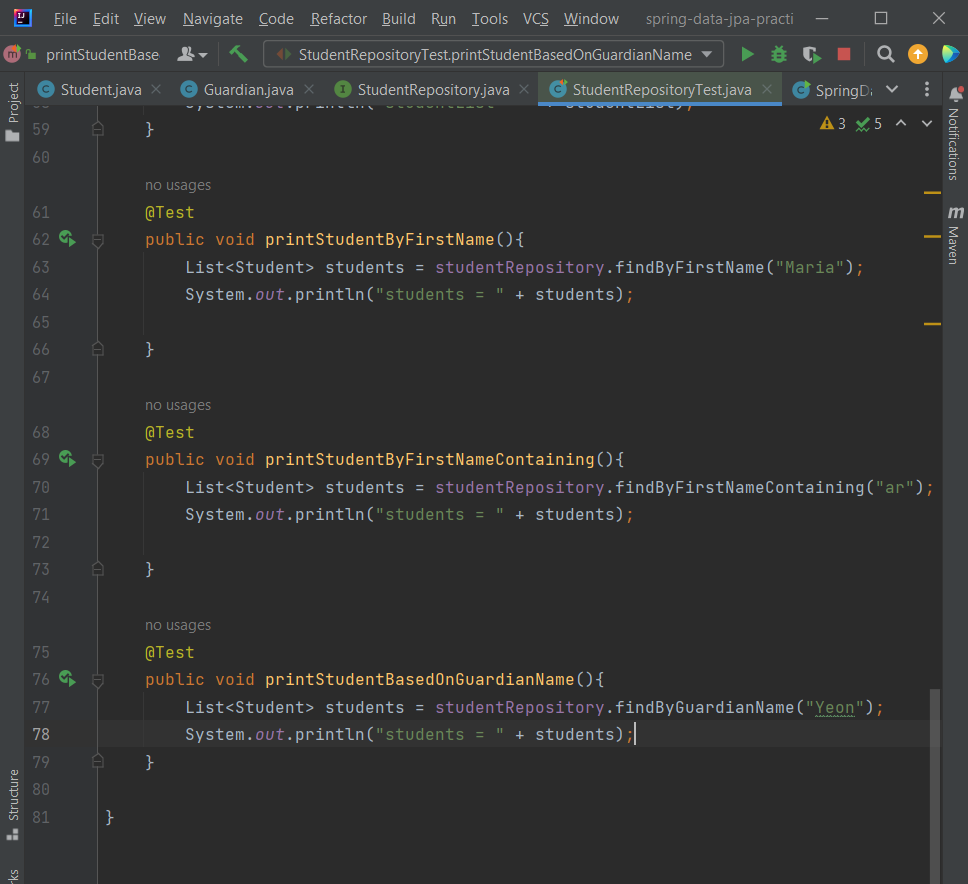
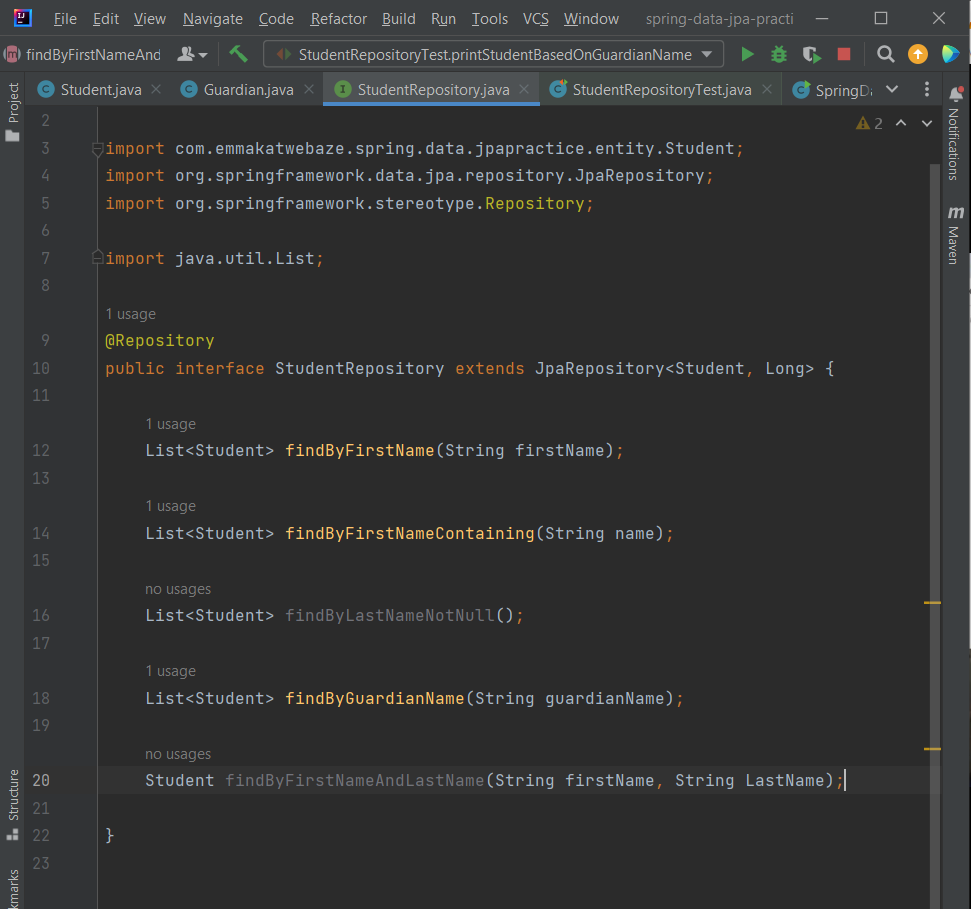
Then change from this;



To @Embedded as below;

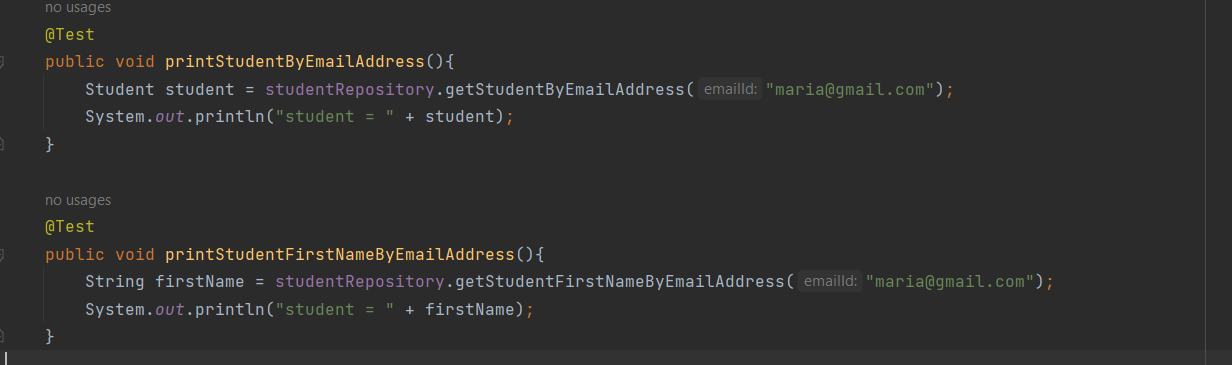
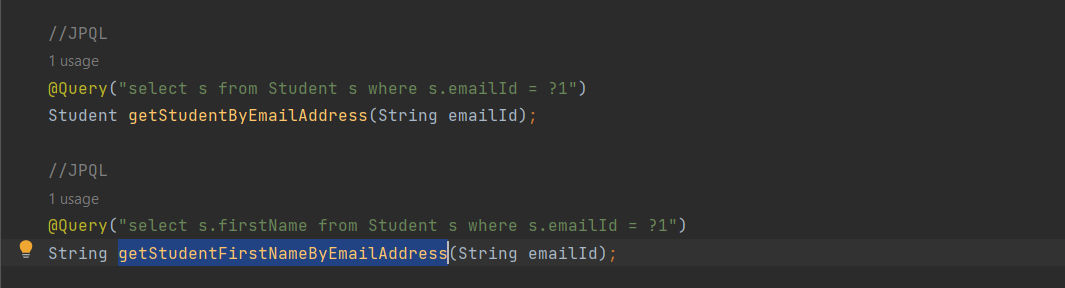


The @AttributeOverrides help us to map this Guardian class data to the Embedded guardian properties all in the student table.

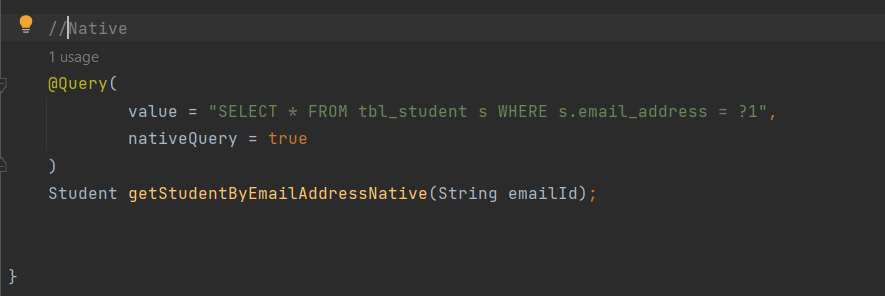
**Creating Custom Jpa Functions**

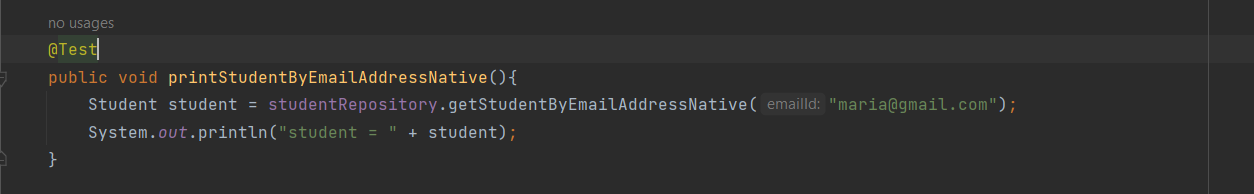
**JPA Queries**

JPA queries are defined based on the classes that you’ve defined not based off the table names or column names.

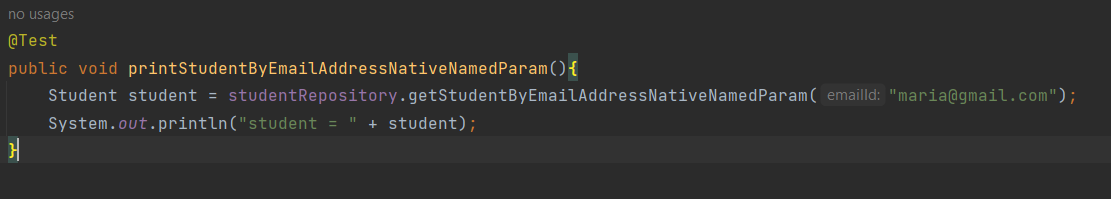
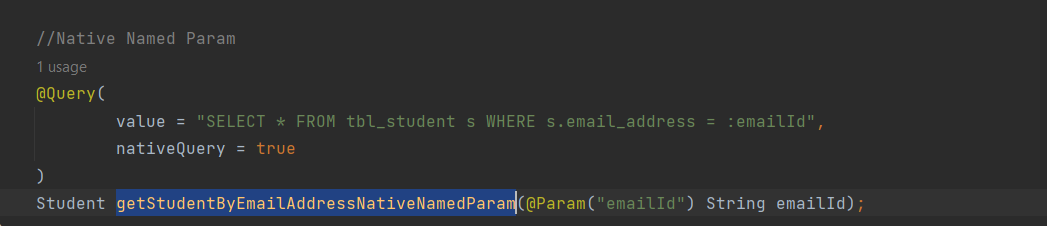


**Native queries include;**

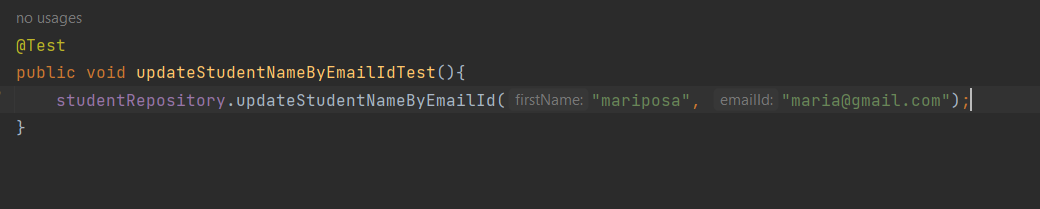
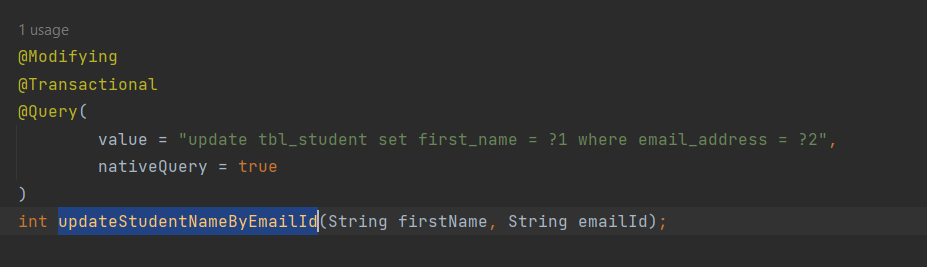
****

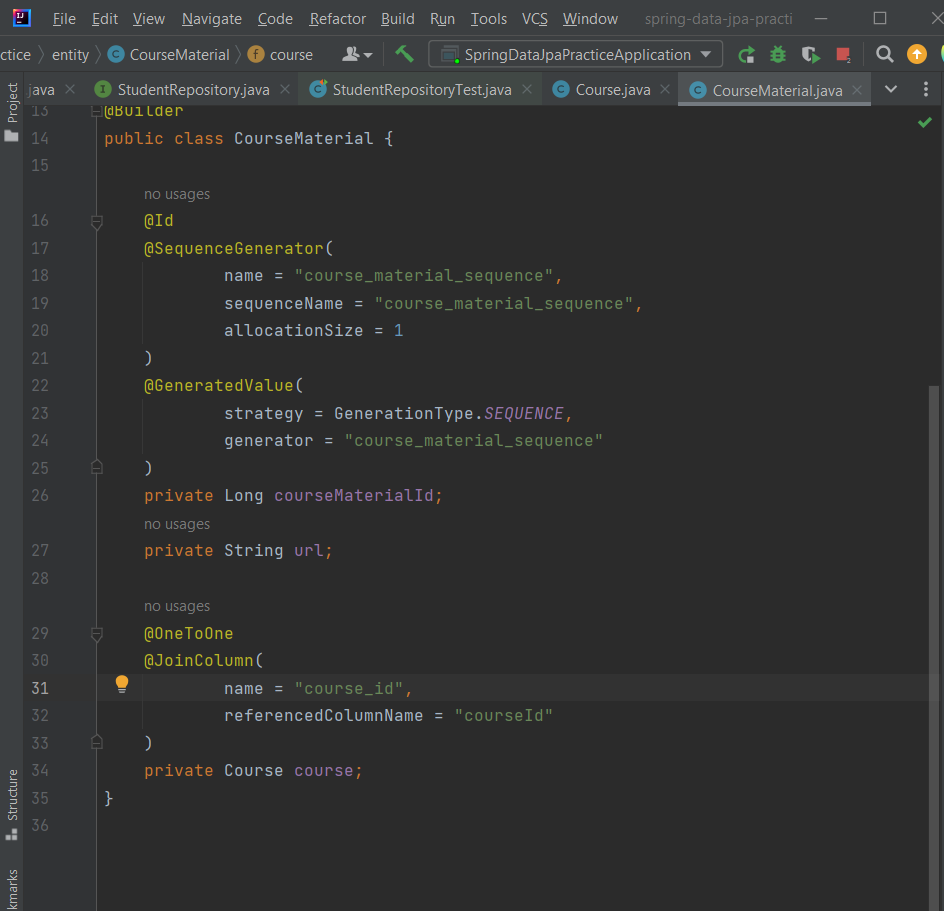
****

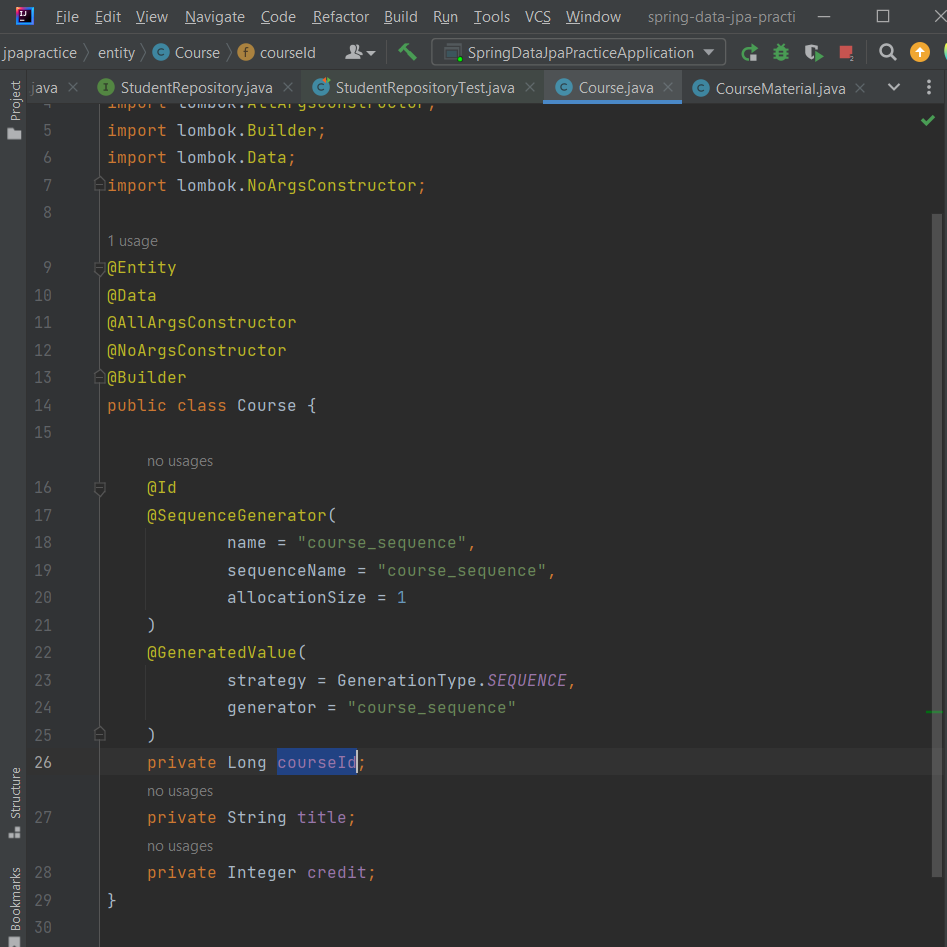
**Native Named Param**

****

**@**Transactional annotation is used to create methods that can only be committed once they are successful and if not they are rolled back. Can also be done in the service layer to make sure that a couple of methods are successful or else if one fails a rollback is done.

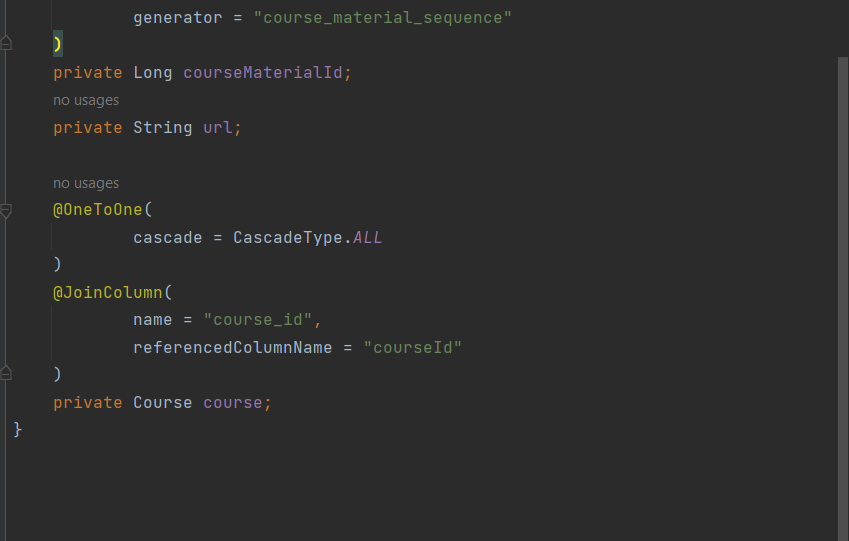
@Modifying is used when you have a function that’s going to modify your records in your database.

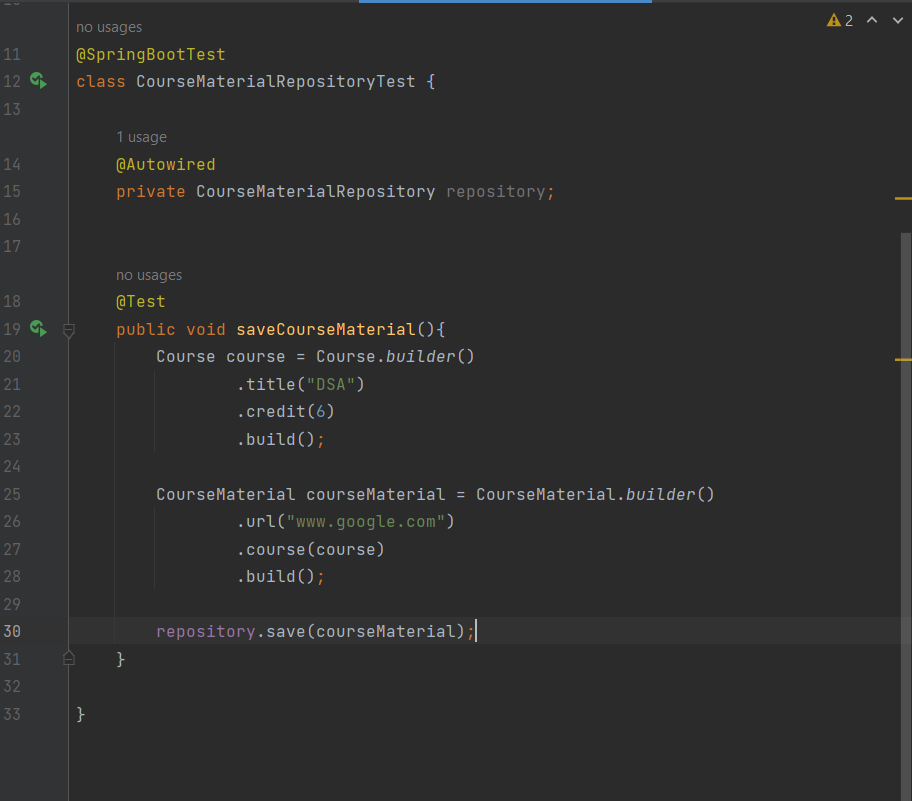
**One to One Relationship in Jpa**

****

**Cascading** means passing rights or information to your child elements.

Meaning you can create a CourseMaterial without a course being existent in the database but at the moment of creating the CourseMaterial, it’s particular course will also be created.

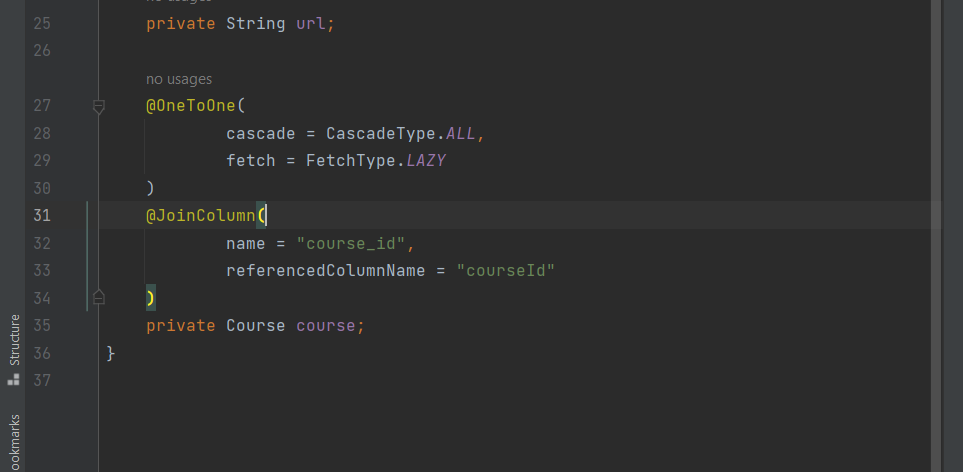


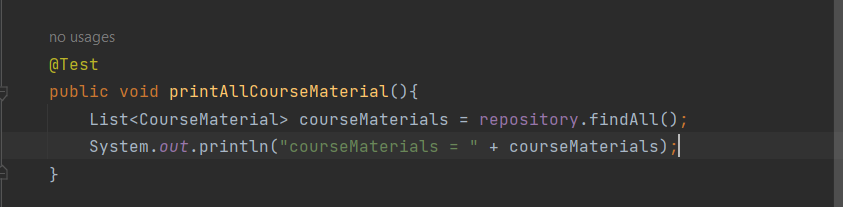


**FetchType** – defines how you want to fetch the data. i.e. Eager fetching and Lazy fetching

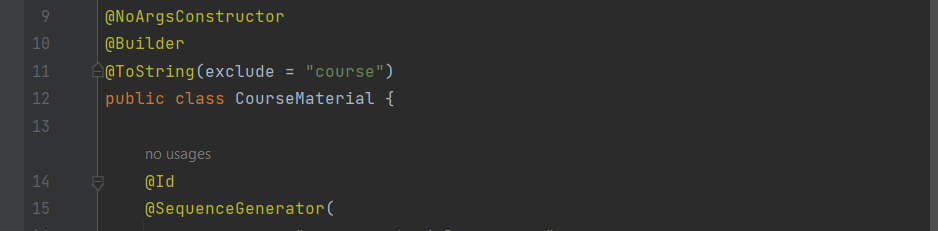
Eager fetching – fetch the course data with course material

Lazy fetching – won’t bring data for course unless you explicitly ask for it.

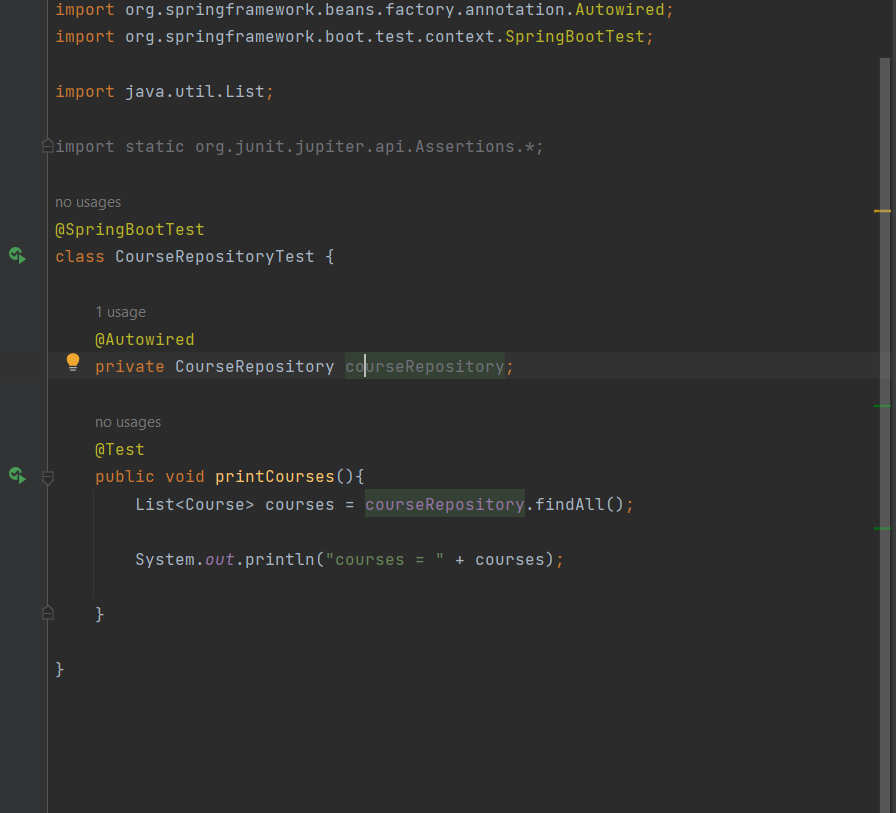
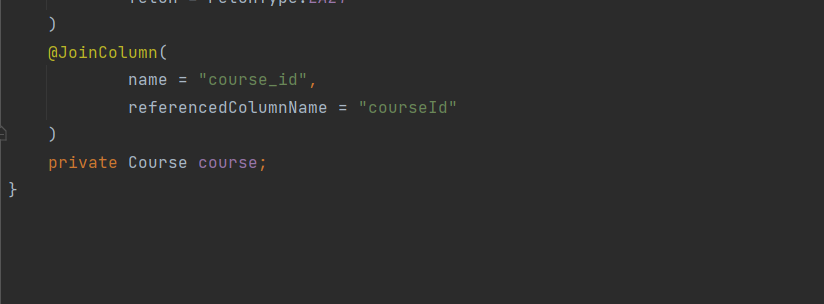
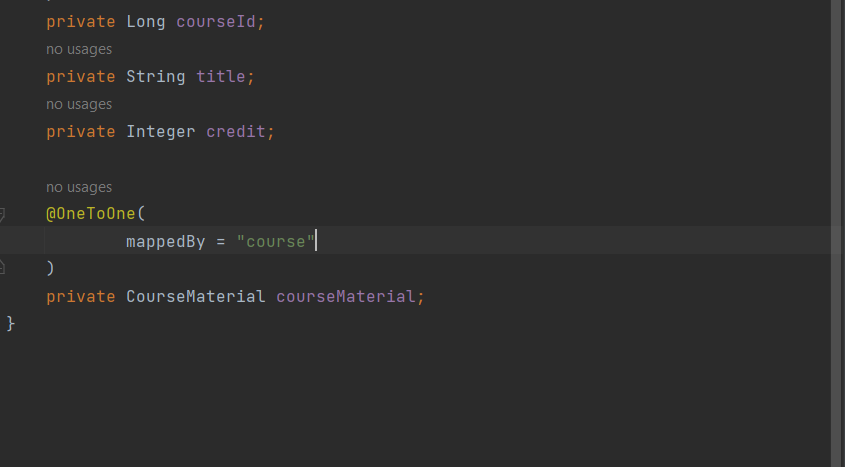




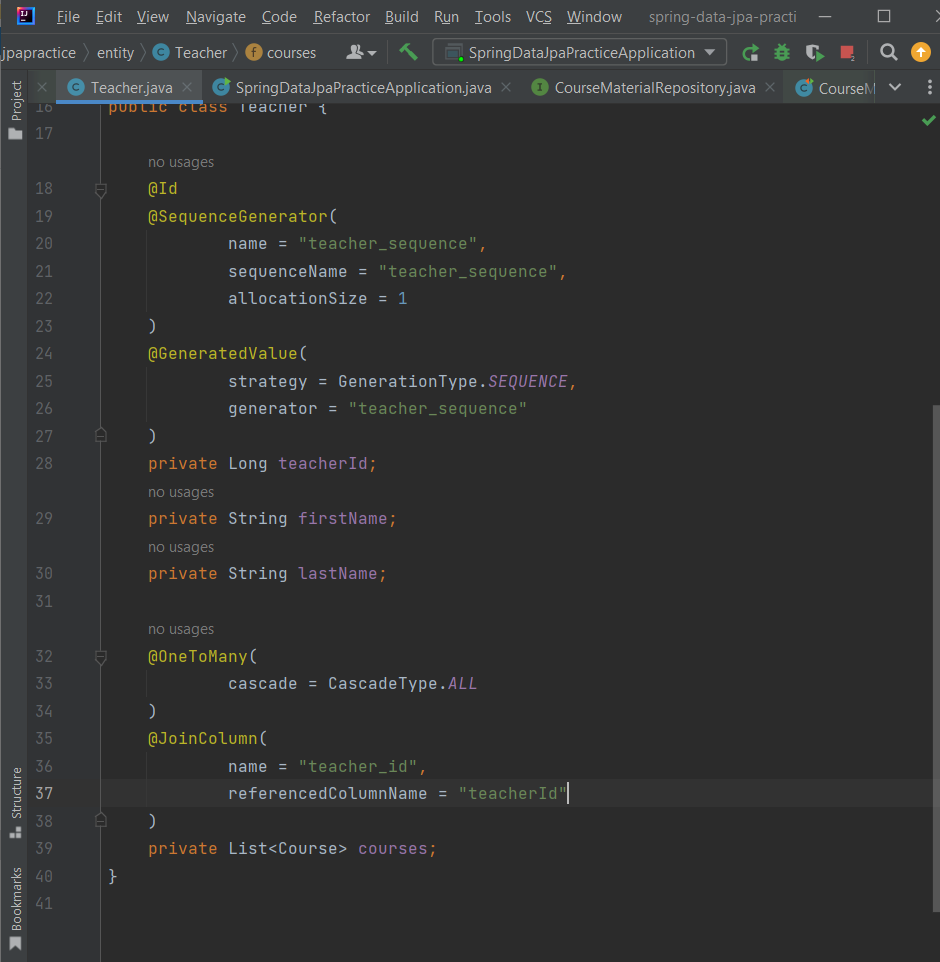
Note: if you’re using Lazy fetching and you’re trying to print out the values make sure you do this;

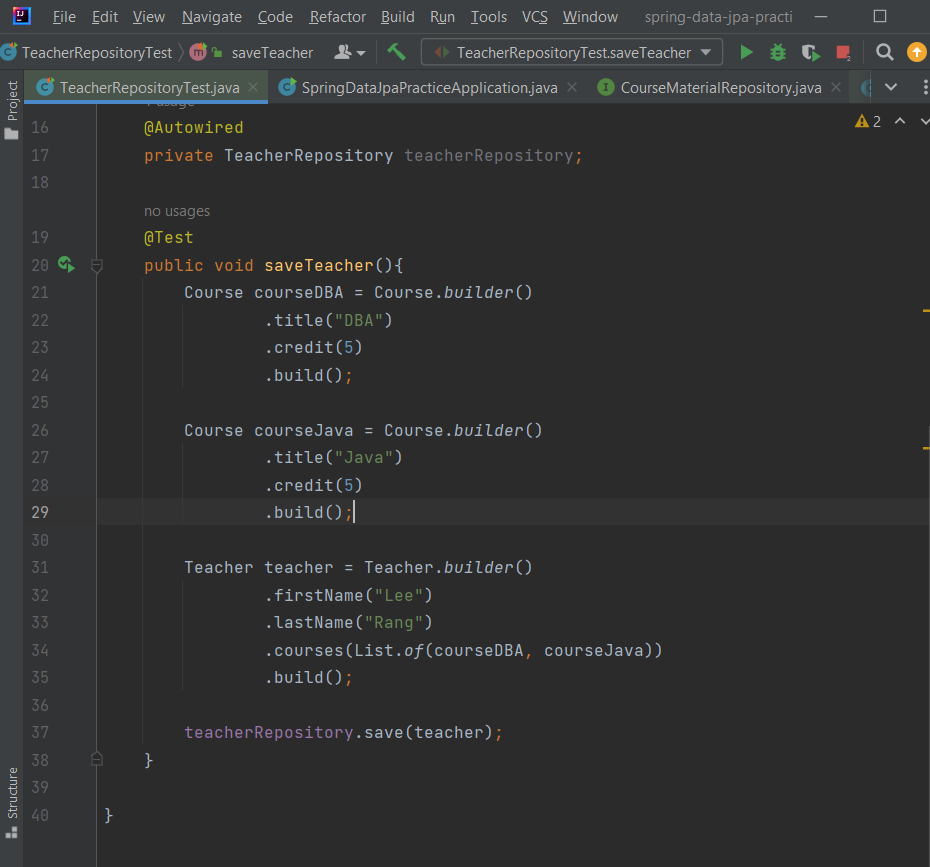


**Bidirectional One to One Mapping**

Basically, you have many instances where you have a foreign key in one table and you don’t have a way to map a certain table to another yet they are linked for example Course doesn’t have courseMaterialId but CourseMaterial has courseId which makes it easy to link to a course.

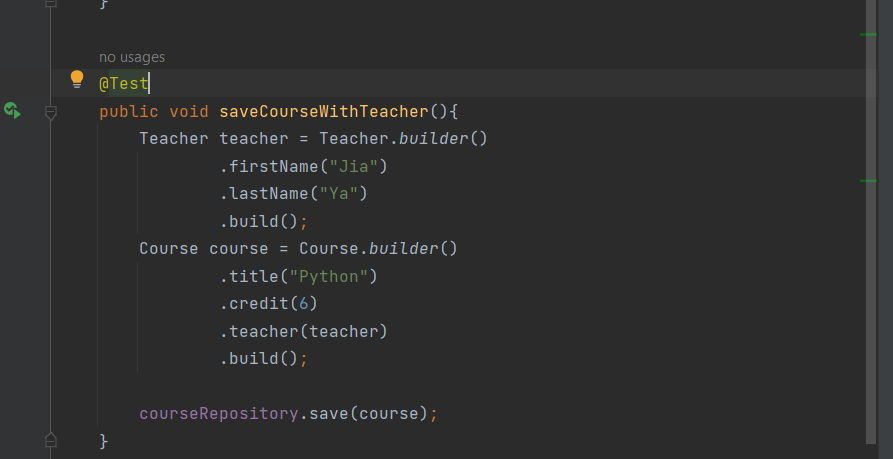
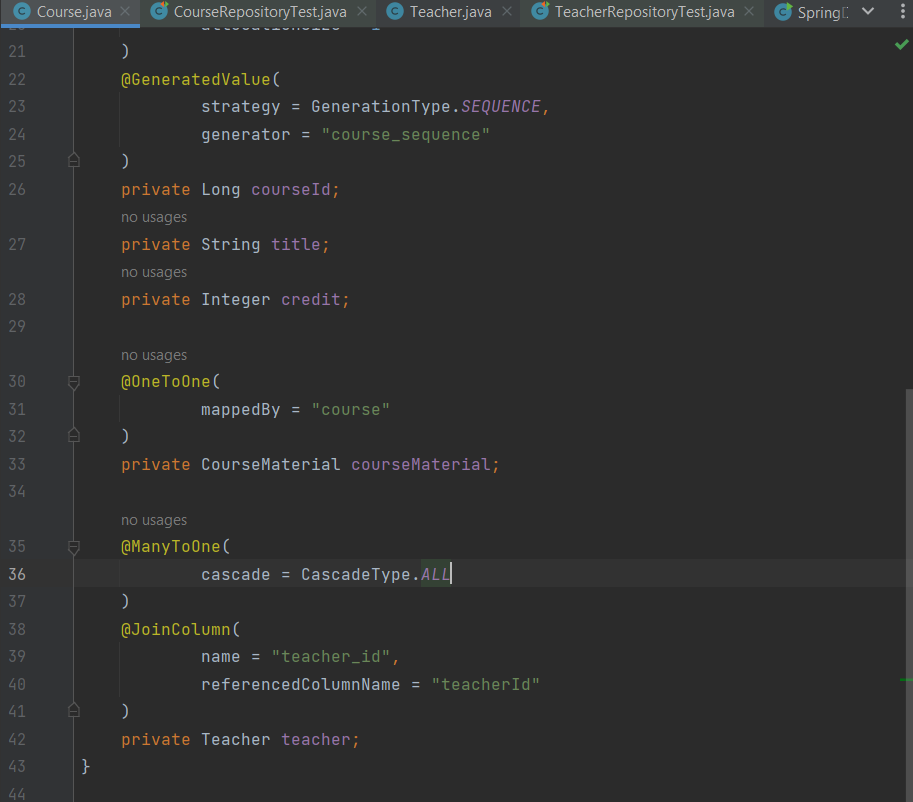
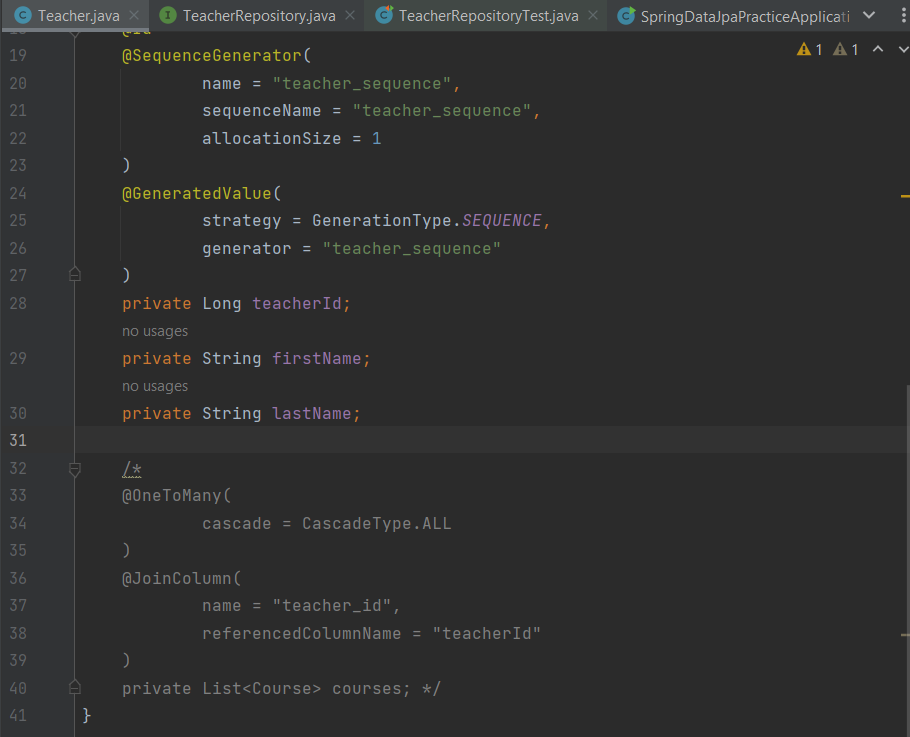
**One to Many Relationship**

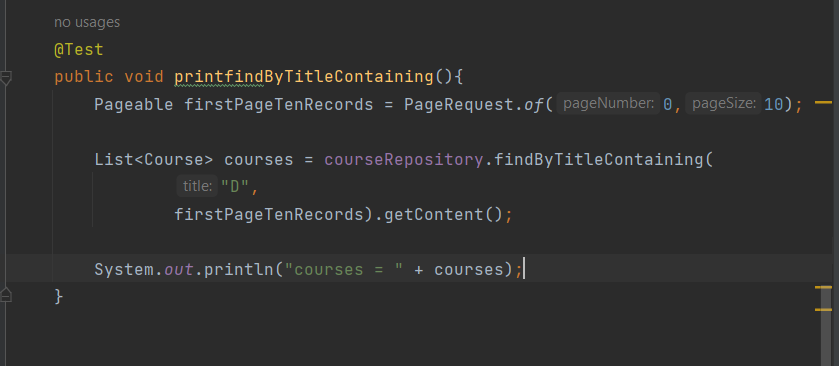
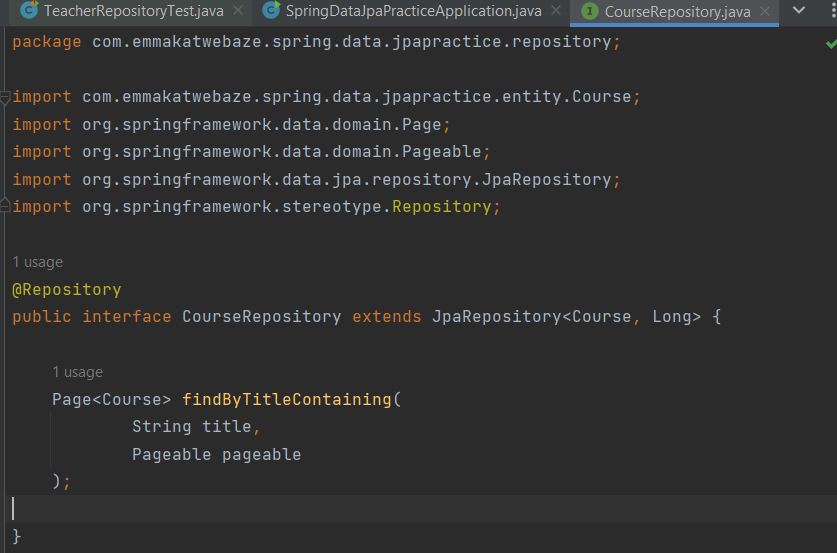
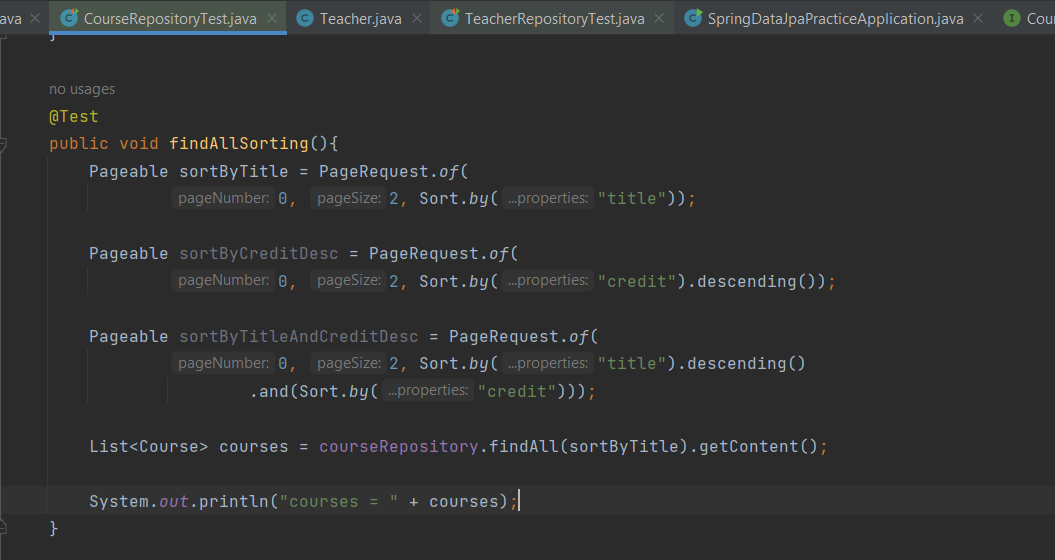
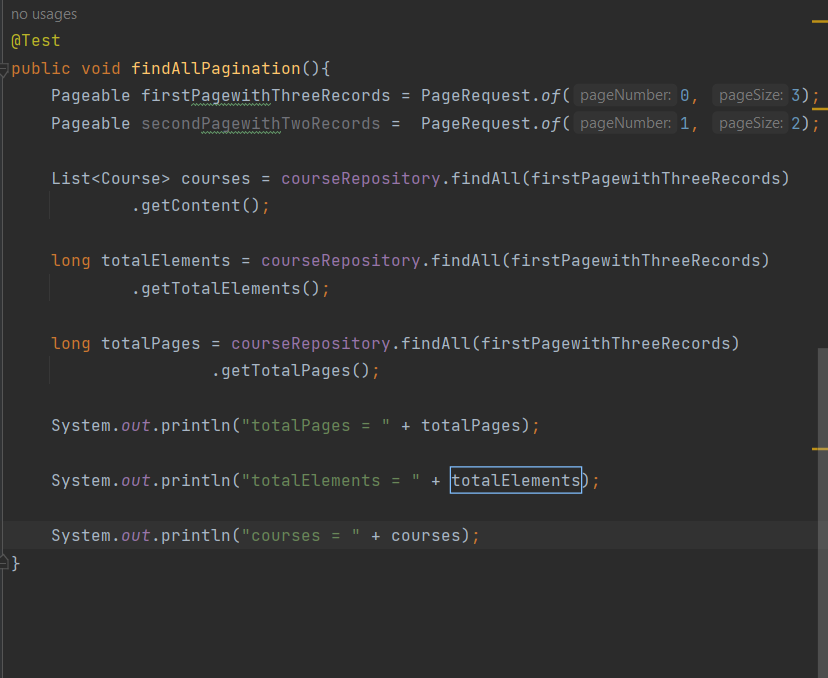
****

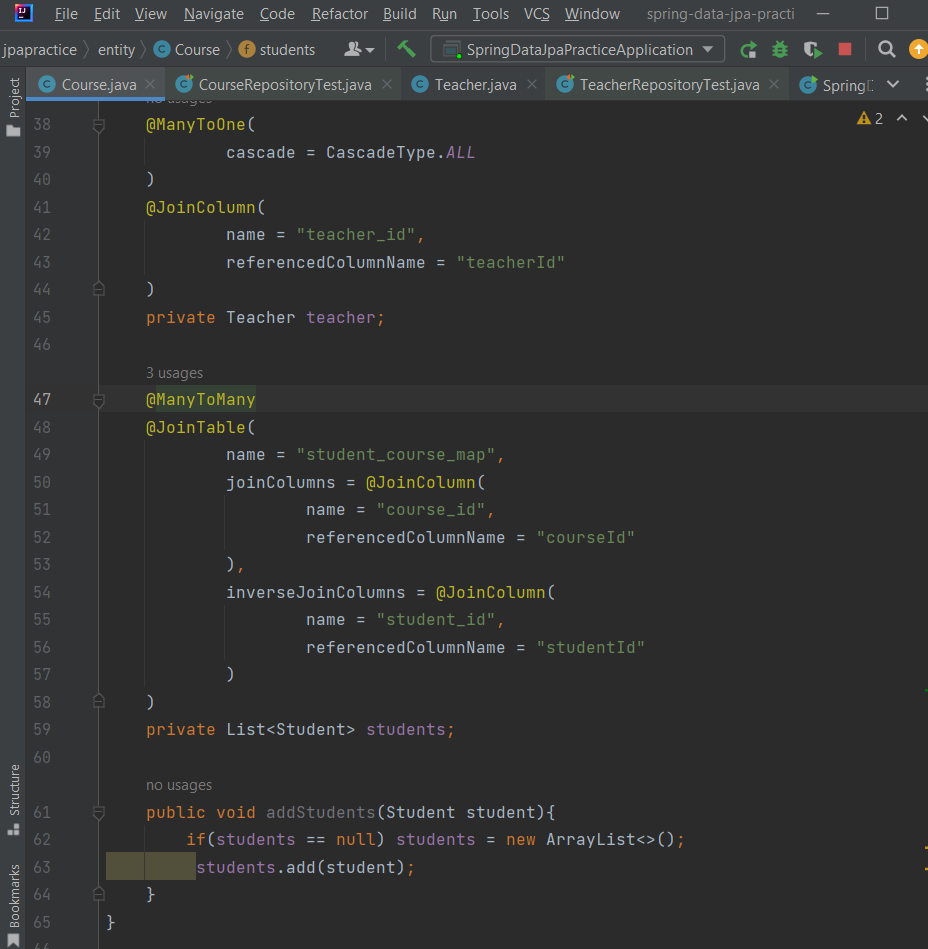


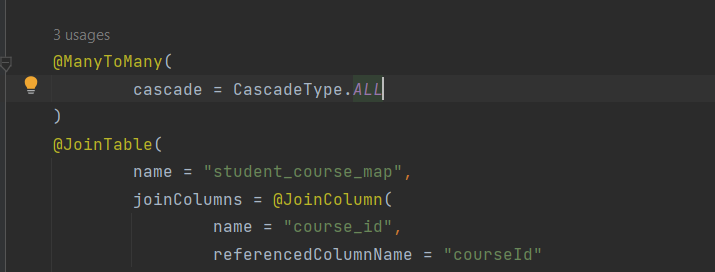
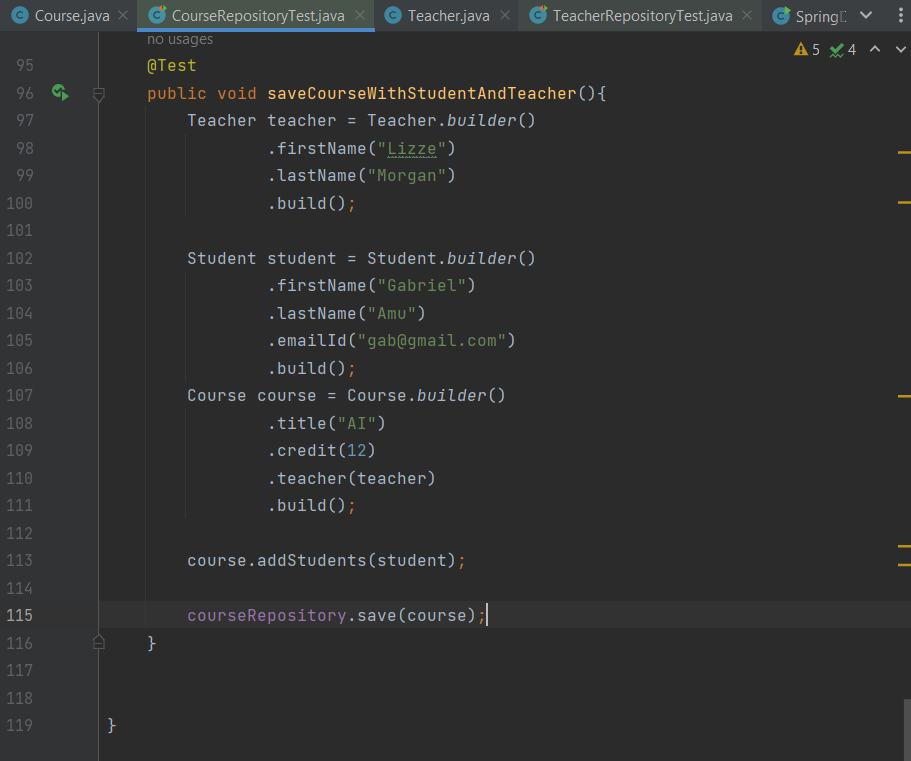
By setting optional = false, it means that whenever you’re trying to save a course, Course Material is required.



**Many to One Relationship**

**Paging and Sorting : use import org.springframework.data.domain.Pageable;**

**Many to Many Relationship**

****