Sring

Creation of the project the dependecies that you have to choose are :

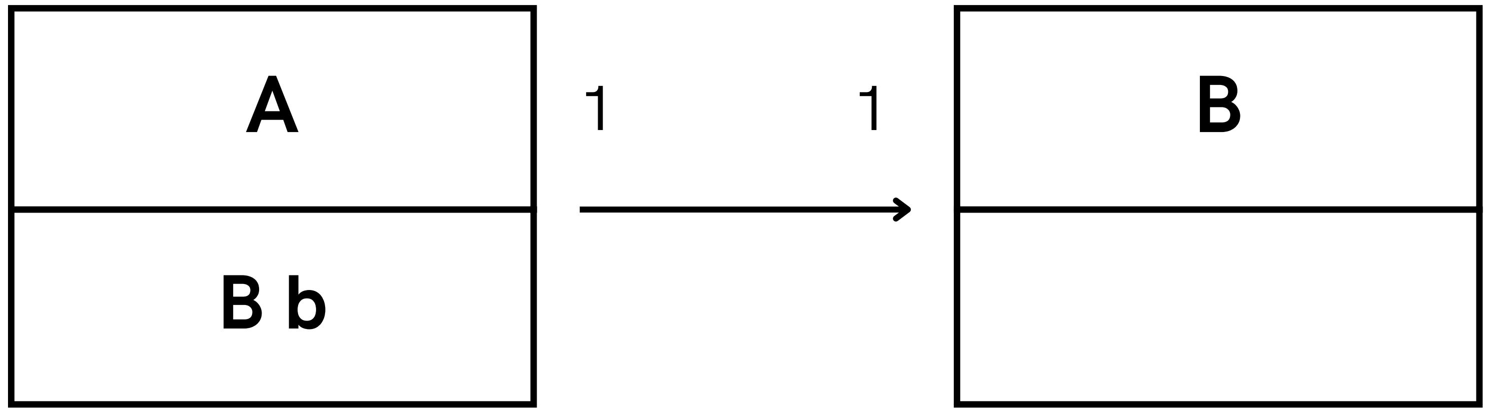
- Lombok
- Web -> SpringWeb
- Spring Data Jpa
- MySQL Driver

Creation of the Packages (Entities, Controllers, Models) :

- For entities : just follow the Structure of the exam
- all the attributs are private (then to got the visibility we have to do the get and the set methods)
- for the Enum (just write as the attributs bellow)
- for calling the Enum in the anothing class we just write like this (Enum Speciality Speciality)

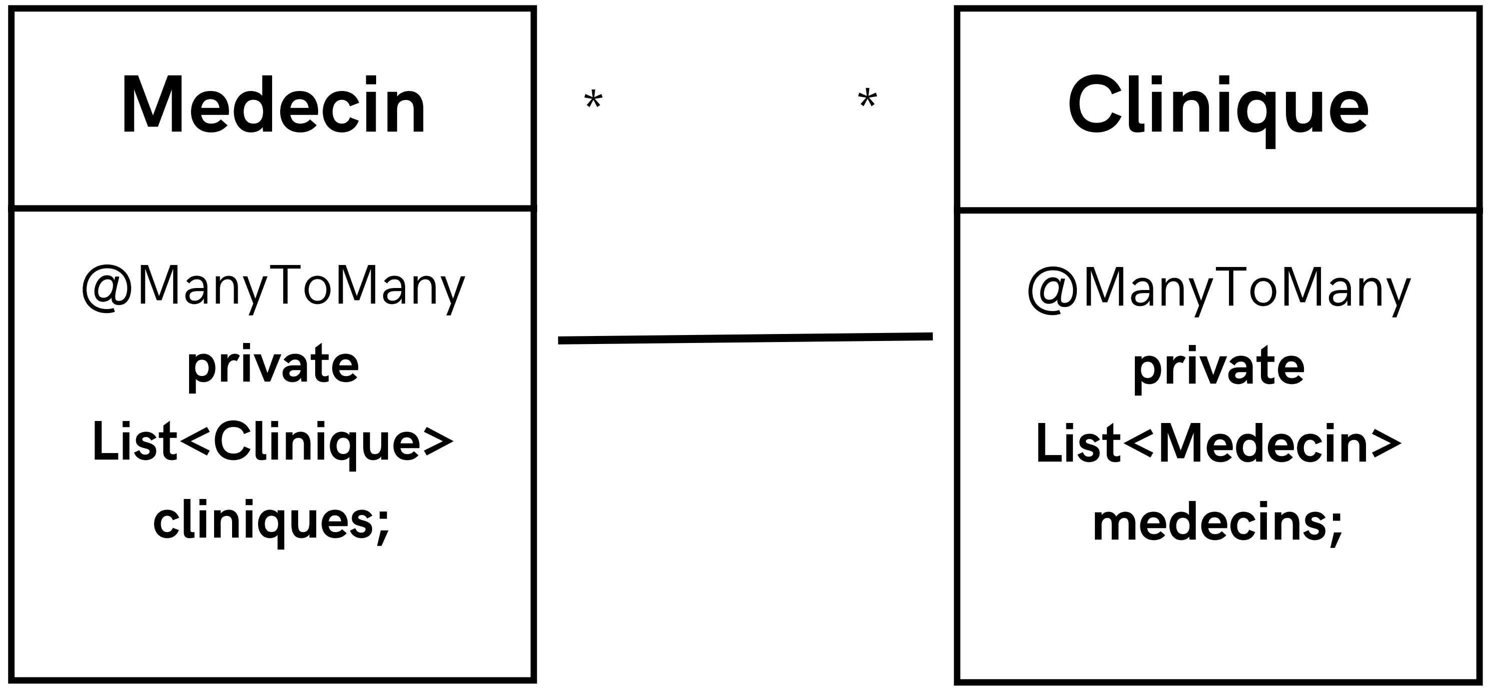
Now the Relations:

- **One to One (unidirectionnel/ bidirectionnel)**
- **One to Many**
- **Many to Many (bidirectionnel/ bidirectionnel)**



- One to One

so the cursor go the way of the B so that means A got the visibility on B But not the same case for B



- Many to Many (bidirectionnel) (PARENT AND CHILD RLS)

in A we call the Attribut that present the B entity and the same case to B
in general "List" Of course do not forget the imports

Now the case in the DataBase :
To relay the entities we need a forgein key (clé primaire)
(9ouwet Spring JPA
for the Good thing we do not need the mention the relationship we just need to call the @

How to get to know who’s the parent and who’s the child :
For the rls
One to Many or Many to One (who’s got the many is the Parent)
But for the One to One and Many to Many you have to guess by logic (and it’s sucks i know)

in our exemple bellow the medecins is the child so the clinique is the parent so to ocnfirm that we code this methode MAPPED BY (<u>in fact if we don’t use Mapped By we can not define the parent/child rls more specefic bidirectionnel</u>)
“Medecin Entity”
@ManyToMany (mapped By = medecins)
private List<Clinique> cliniques;

so moulakhes lahkeya kn talka flesh blesh curseur fil diagramme de classe aref rahou hethika cas mtaa directionnel donc nahkiw toul aala relation parent/fils ken l rls manytoone wala onetomany aref li aandou many houwa l parent w ken ManytoMany wala OneToOne rahou besh tarref bil logique mteeik wala ikolik fil ennoncé w besh nkoulou chkoun l child fil code nestamlou “mappedBy”
taw kn flesh bil curseur rana nahkiw aala relation uniderctionnel illi aaliha w hkina aaliha fil diapo li kablou
:)

For the importations
we got @Getter and @Setter : so we dont have to wrote to full codes
@NoArgContrustor (constructeur feragh)
@AllArgContrustor (constructeur)
@ToString(ihawel interger l string)
@Entity (ki nhebou yasn3el tableau fil base de donnée donc lezemna l nzidou primary key)
@Id (lil primary key)
@GeneratedValue (strategy = GenerationType.IDENTITY) "for render it autoincrement"

For connection to Database we got to
applicaition.properties file and we add those lines
spring.datasource.url=jdbc:mysql://localhost:3306/medecin?createDataBaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update (update database kol matzid haja jdida)

to add repositories we gonna need to add this line over here <nameoftheclass,type of the id >

```
public interface RendezVousRepository extends JpaRepository <RendezVous,Long> {
```

Now for service (interface for faible couplage)

for serviceExamenImpl

```
package tn.esprit.medecin.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import tn.esprit.medecin.entity.Clinique;
import tn.esprit.medecin.repository.CliniqueRepository;
import tn.esprit.medecin.repository.MedecinRepository;
import tn.esprit.medecin.repository.PatientRepository;
import tn.esprit.medecin.repository.RendezVousRepository;

@Service
public class ServiceExamenImpl implements ServiceExamen {
```

```
@Autowired
 private CliniqueRepository cliniqueRepository;
 @Autowired
 private MedecinRepository medecinRepository;
 @Autowired
 private PatientRepository patientRepository;
 @Autowired
 private RendezVousRepository rendezVousRepository;
 @Override
 public Clinique addClinique(Clinique clinique) {
 return cliniqueRepository.save(clinique);(.save to add)
 }
}
```

For affectaion now w e can and affect to it at the same time (add a doctor and effecte to clinique)

for case Many to Many

```
@Override
public Medecin addMedecinAndAssignToClinique(Medecin medecin, Long IdClinique) {
    //awel lawej par id clinique kn medecin mawjouda wala le
Clinique c = cliniqueRepository.findById(IdClinique).orElse(null);
    List<Medecin> list = new ArrayList<>();
    list.add(medecin);
   ://in case mafamesh nzidouh
 if (c.getMedecins() == null) {
        c.setMedecins(list);
    } else {
        c.getMedecins().add(medecin);
    }
    return medecinRepository.save(medecin);
}}
```

For now case One to Many (rendezvous is the parent)

```
@Override
public RendezVous addRDVandAssignMedAndPatient(RendezVous rendezVous, Long idMedecin, Long
idPatient) {
 Medecin m = medecinRepository.findById(idMedecin).orElse(null);
 Patient p = patientRepository.findById(idPatient).orElse(null);
 rendezVous.setPatient(p);
 rendezVous.setMed(m);
 return rendezVousRepository.save(rendezVous);
}
```

For schelure :

AOP :
besh nkasmou l parte logique de code