ORM associaties: relaties

Veerle Ongenae

Relaties

- Value-objecten
- Relaties tussen entiteiten

Mapping van associaties van entiteiten

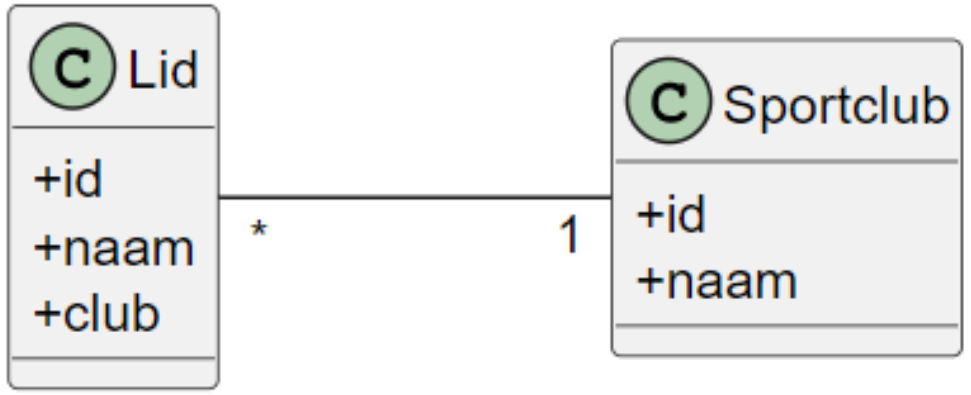
UNIVERSITEIT GENT

- 1-1
  - Unidirectioneel
  - Bidirectioneel
- 1- veel
  - Unidirectioneel
  - Bidirectioneel
- Veel – veel
  - Unidirectioneel
  - Bidirectioneel

1- veel unidirectioneel

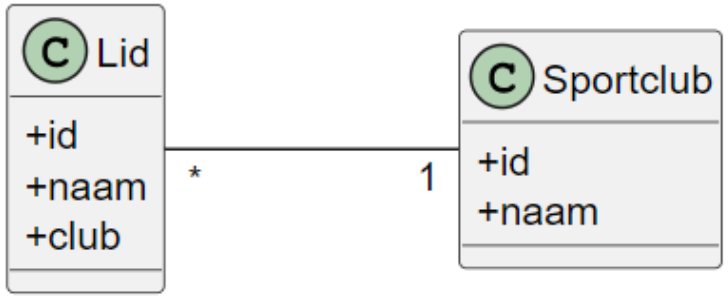
- Met foreign key

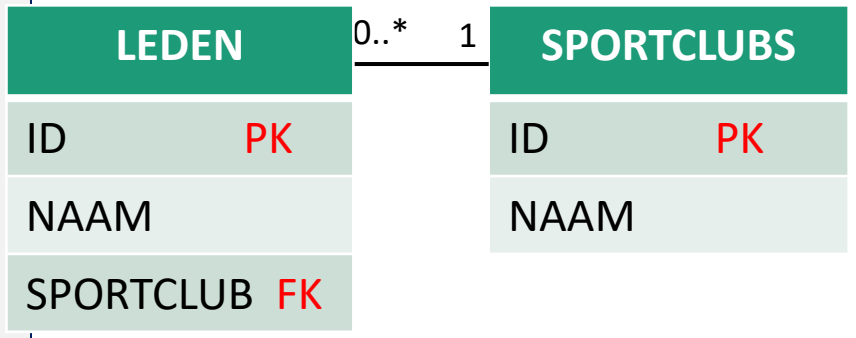
LEDEN		0..*	1	SPORTCLUBS	
ID	PK			ID	PK
NAAM				NAAM	
SPORTCLUB	FK				



1- veel unidirectioneel: foreign key

```
@Entity
@Table(name = "leden")
public class Lid implements Serializable {
    …
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public int getId() {…}
    …
    @ManyToOne
    @JoinColumn(name="sportclub")
    public Sportclub getClub() {…} …}
```





1-veel unidirectioneel: foreign key

```
@ManyToOne
@JoinColumn(name="sportclub")
```

- Veel-1 relatie
  - Veel leden voor één sportclub
  - Naam kolom met verwijssleutel naar geassocieerde entiteit
    - Optioneel indien zelfde als naam eigenschap
  - Attribuut `nullable` (`true`/`false`)
    - Verplicht veld?

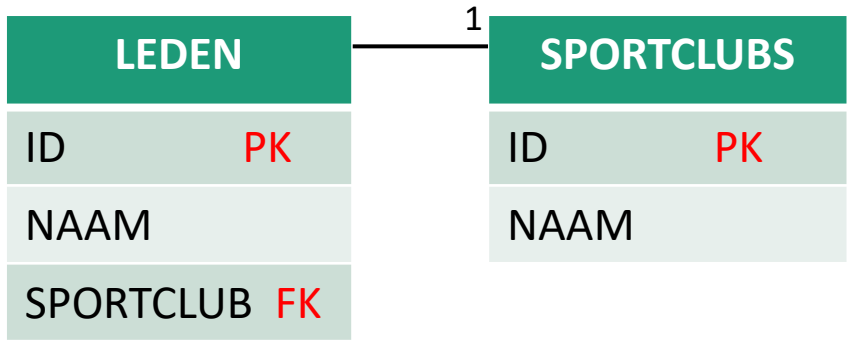
Mapping van associaties van entiteiten

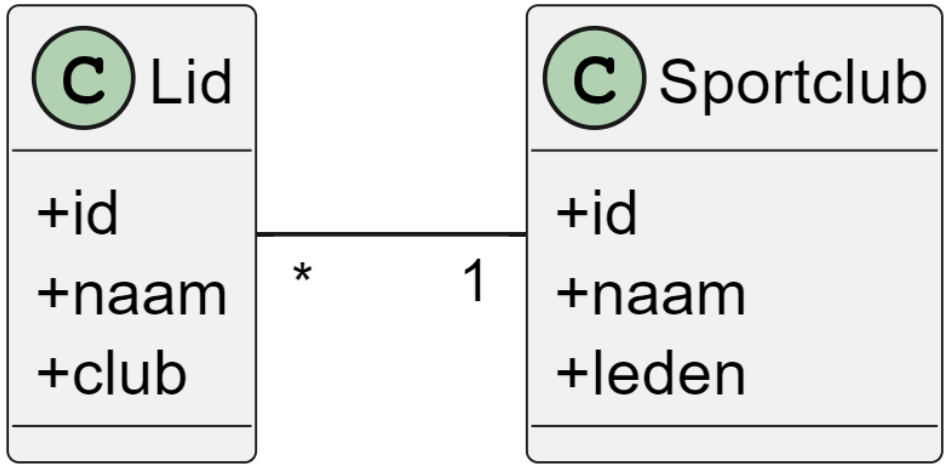
UNIVERSITEIT GENT

- 1-1
  - Unidirectioneel
  - Bidirectioneel
- 1- veel
  - Unidirectioneel
  - Bidirectioneel
- Veel – veel
  - Unidirectioneel
  - Bidirectioneel

1- veel bidirectioneel

- Met foreign key





Bidirectioneel in code

```
Sportclub maakSportclubMetLeden(String naam) {
    Sportclub club = maakSportclub(naam);
    club.setNaam(naam);
    club.setLeden(maakLeden());
    club.getLeden().forEach(lid -> lid.setClub(club));
    return club;
}
```

```
// verander van club
Lid lid = club1.getLeden().get(index);
lid.setClub(club2);
club2.getLeden().add(lid);
club1.getLeden().remove(lid);
```

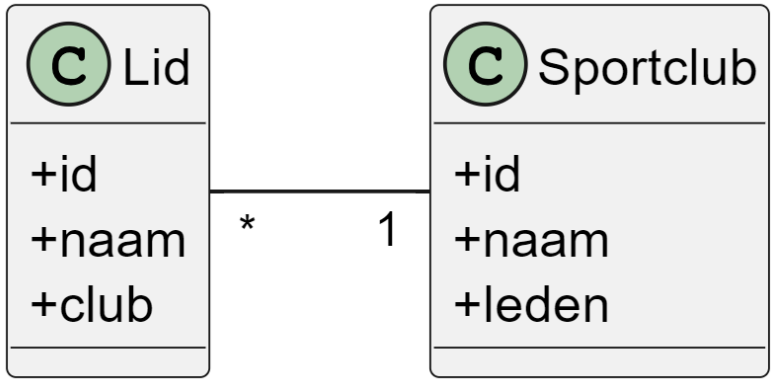
Aanpassing annotatie Sportclub

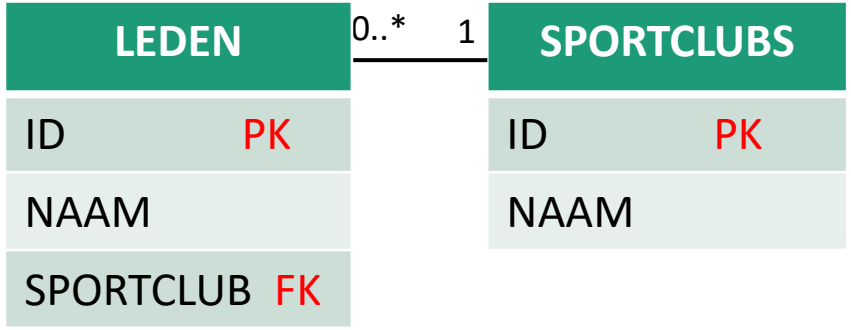
Lid

```
@ManyToOne
@JoinColumn(name="sportclub")
public Sportclub getClub() {
  return club;
}
```

Sportclub

```
@OneToMany(mappedBy="club")
public Set<Lid> getLeden() {
  return leden;
}
```





1-veel bidirectioneel: foreign key

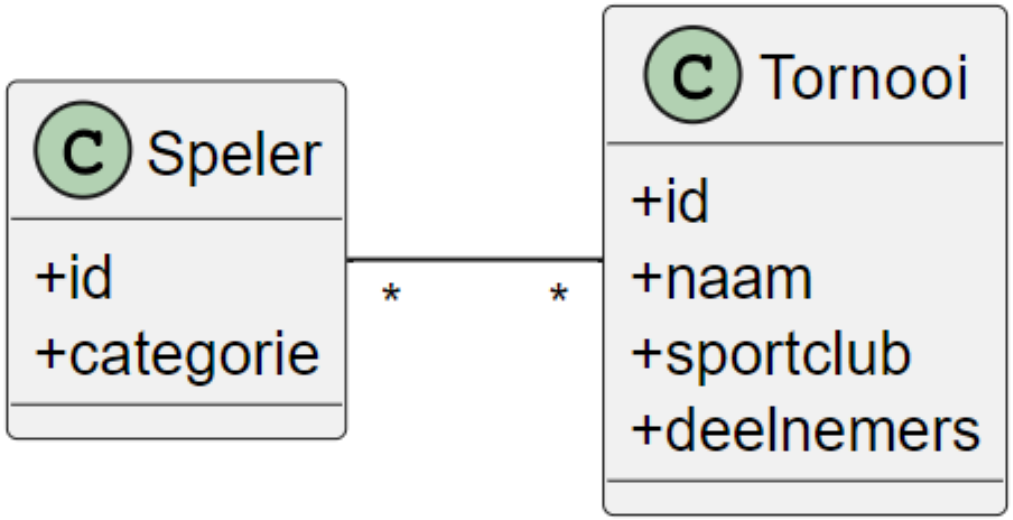
```
@OneToMany(mappedBy="club")
public Set<Lid> getLeden() {
  return leden;
}
```

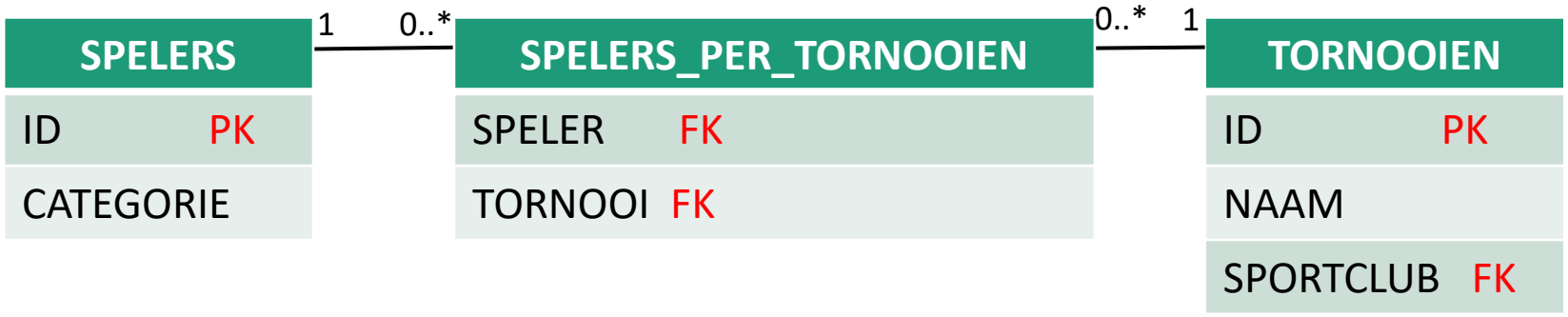
- 1-veel relatie
- Naam van de eigenschap van Lid die de relatie beschrijft
- Altijd bij @OneToMany niet bij @ManyToOne

Mapping van associaties van entiteiten

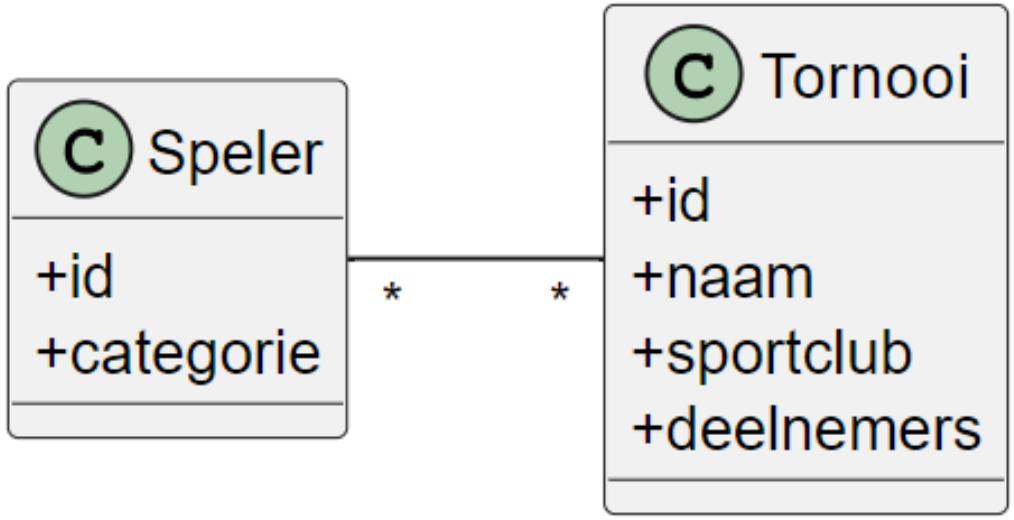
- 1-1
  - Unidirectioneel
  - Bidirectioneel
- 1- veel
  - Unidirectioneel
  - Bidirectioneel
- Veel – veel
  - Unidirectioneel
  - Bidirectioneel

Veel-veel unidirectioneel





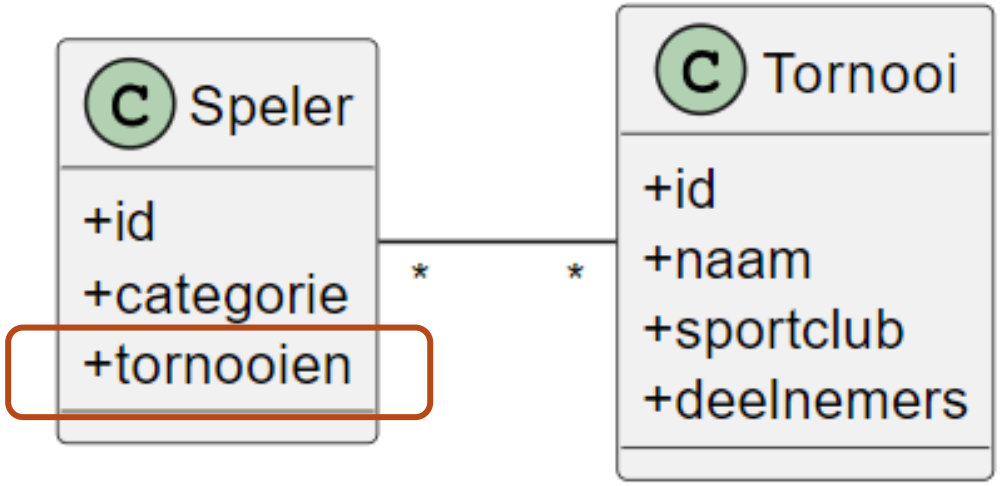
Annotatie

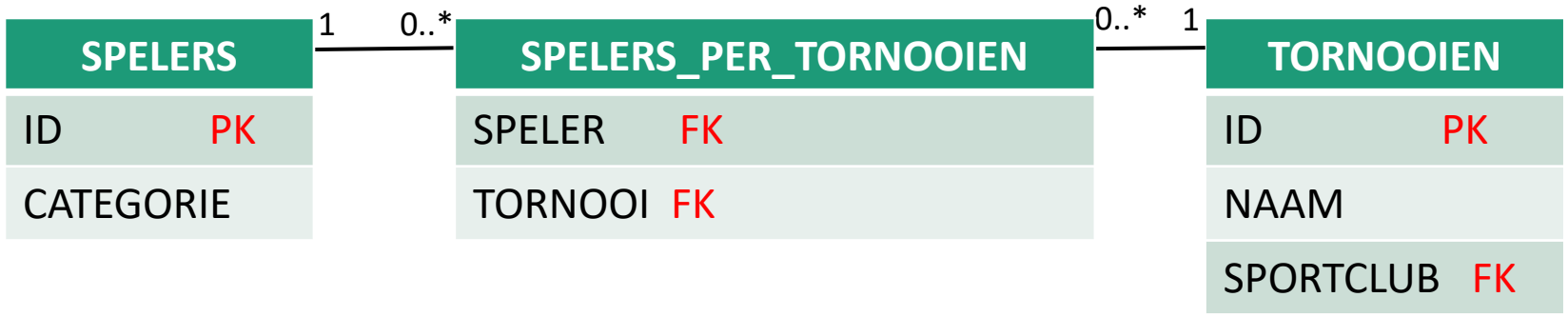



```
@ManyToMany
@JoinTable(name="SPELERS_PER_TORNOOI",
   joinColumns=@JoinColumn(name="tornooi"),
   inverseJoinColumns=@JoinColumn(name="speler"))
```

- Veel-veel relatie
- Associatietabel
  - Naam van de associatietabel
  - Naam kolom met verwijssleutel naar deze entiteit
  - Naam kolom met verwijssleutel naar element in collectie

Veel-veel bidirectioneel





Aanpassingen annotaties

Speler:

```
@ManyToMany(mappedBy="deelnemers")
public Set<Tornooi> getTornooien() {
  return tornooien;
}
```

- Veel-veel relatie
- Naam van de eigenschap van Tornooi die de relatie beschrijft

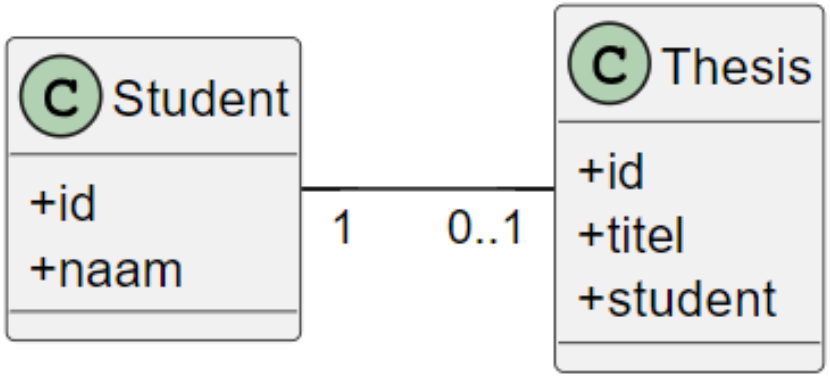
Mapping van associaties van entiteiten

- 1-1
  - Unidirectioneel
  - Bidirectioneel
- 1- veel
  - Unidirectioneel
  - Bidirectioneel
- Veel – veel
  - Unidirectioneel
  - Bidirectioneel

1-1 unidirectioneel

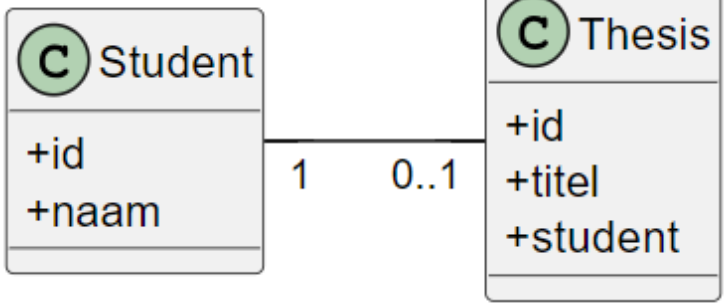
- Beide entiteiten delen dezelfde primary key

STUDENTEN	1	0..1	THESISSEN
ID PK			ID PK FK
NAAM			TITEL
…			



- Een van beide entiteiten heeft een foreign key (met unique constraint)

STUDENTEN	1	0..1	MASTERPROEF
ID PK			ID PK
NAAM			STUDENT FK
…			TITEL



STUDENTEN	
ID	PK
NAAM	
…	

1 0..1

THESISSEN	
ID	PK FK
TITEL	

```
@Entity
@Table(name = "studenten")
public class Student implements Serializable {
    private int id;
    private String naam;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public int getId() {…}
…}
```

```
@Entity
@Table(name="THESISSEN")
public class Thesis implements Serializable {
    private int id;
    private String titel; private Student student;
    @Id()
    public int getId() {…}
    @OneToOne()
    @MapsId
    @JoinColumn(name="id")
    public Student getStudent() {…}
…}
```

Beide entiteiten delen dezelfde primary key: annotaties

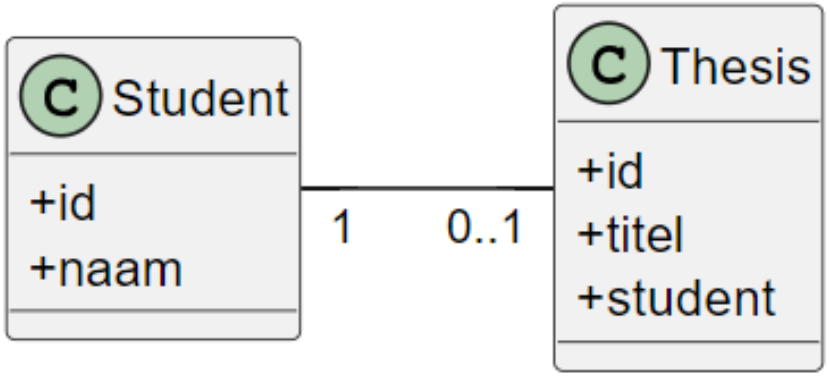
```
@Id()
public int getId() {…}
@OneToOne()
@MapsId
@JoinColumn(name="id")
public Student getStudent() {…}
```

- Geen gegenereerde waarde voor de id
- 1-1 relatie
  - Gebruik de primaire sleutel van de geassocieerde entiteit
  - Kolomnaam voor de primaire sleutel

1-1 unidirectioneel

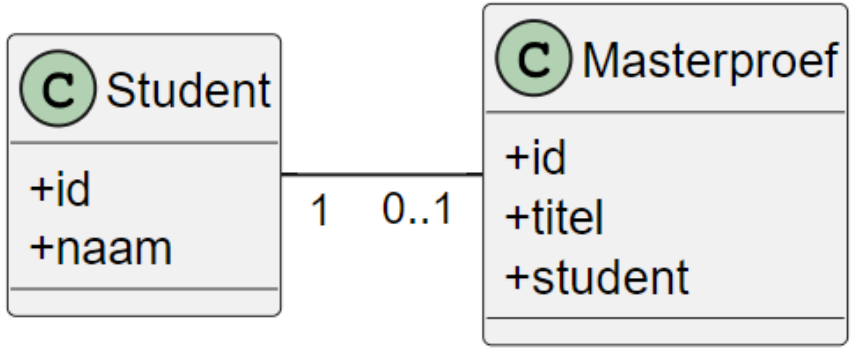
- Beide entiteiten delen dezelfde primary key

STUDENTEN	1	0..1	THESISSEN
ID PK			ID PK FK
NAAM			TITEL
…			

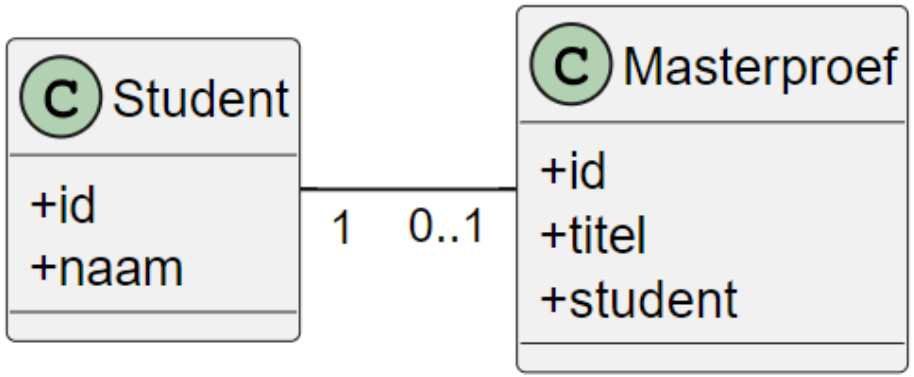


- Een van beide entiteiten heeft een foreign key (met unique constraint)

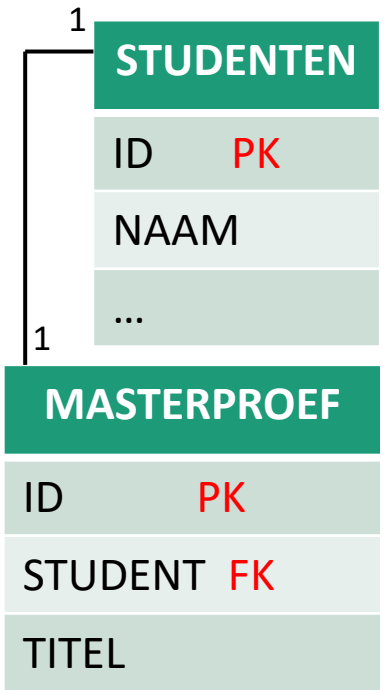
STUDENTEN	1	0..1	MASTERPROEF
ID PK			ID PK
NAAM			STUDENT FK
…			TITEL



```
@Entity
@Table(name = "studenten")
public class Student implements Serializable {
  …
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public int getId() {…}
…}
```




```
@Entity
public class Masterproef implements Serializable {
  …
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public int getId() {…}
  @OneToOne(optional=false)
  @JoinColumn(name="student",unique=true,nullable=false,updatable=false)
  public Student getStudent() {…}
…}
```



1-1 unidirectioneel: annotaties

```
@OneToOne(optional=false)
@JoinColumn(name="student", unique=true,
  nullable=false, updatable=false)
public Student getStudent() {…}
```

- 1-1 relatie, kan niet null zijn
- Kolom met verwijssleutel
  - Kolomnaam
  - Kenmerken primaire sleutel

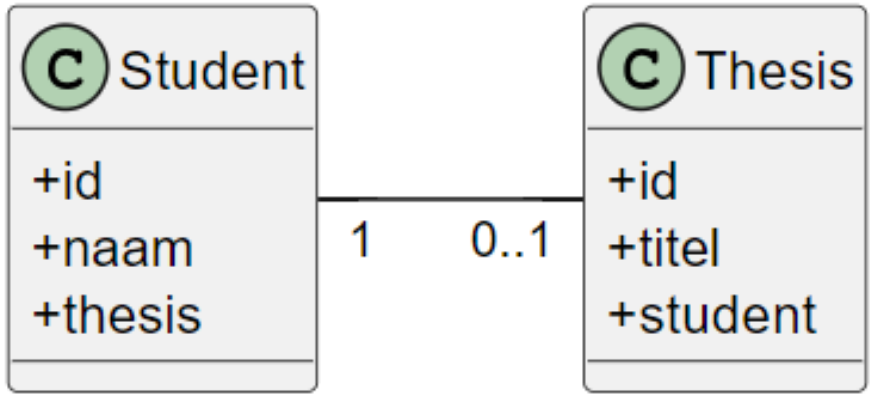
Mapping van associaties van entiteiten

- 1-1
  - Unidirectioneel
  - Bidirectioneel
- 1- veel
  - Unidirectioneel
  - Bidirectioneel
- Veel – veel
  - Unidirectioneel
  - Bidirectioneel

1-1 bidirectioneel

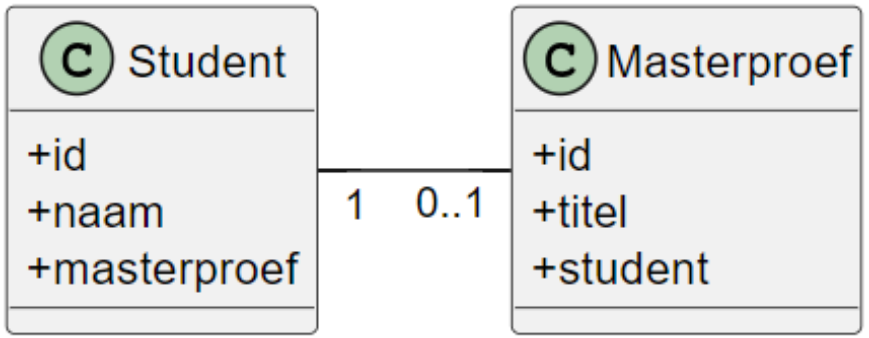
- Beide entiteiten delen dezelfde primary key

STUDENTEN	1	0..1	THESISSEN
ID PK			ID PK FK
NAAM			TITEL
…			



- Een van beide entiteiten heeft een foreign key (met unique constraint)

STUDENTEN	1	0..1	MASTERPROEF
ID PK			ID PK
NAAM			STUDENT FK
…			TITEL



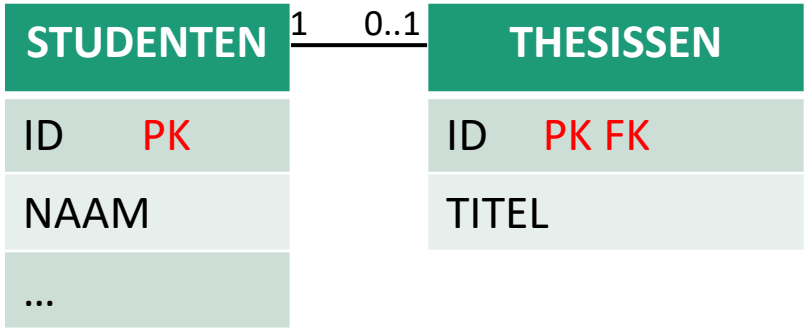
Aanpassen klasse Student en Masterproef

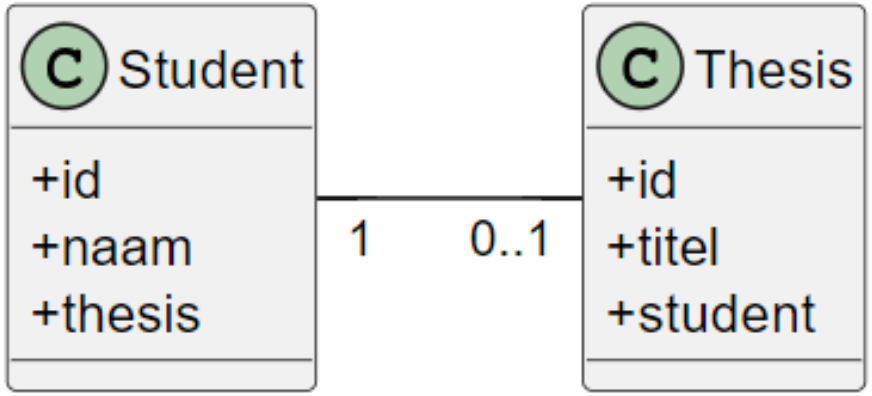
- Eigenschap masterproef toevoegen aan Student
- Eenduidigheid toevoegen

```
List<Masterproef> maakMasterproeven(List<Student> studenten) {
     List<Masterproef> masterproeven = new ArrayList<>();
     String[] titels = {"Hibernate", "ORM", "Linq", "JDBC",
        "ADO.NET", "JSF", "JPA", "JAXB", "Webservices"};
     for (int i = 0; i < titels.length; i++) {
        Masterproef masterproef = new Masterproef();
        masterproef.setTitel(titels[i]);
        masterproef.setStudent(studenten.get(i));
        studenten.get(i).setMasterproef(masterproef);
        masterproeven.add(masterproef);
    }
    return masterproeven;
}
```

1-1 bidirectioneel: annotaties

- Beide entiteiten delen dezelfde primary key





- Thesis

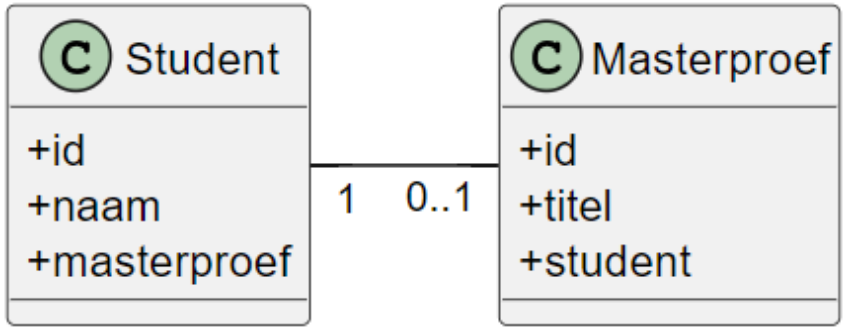
```
@Id() public int getId() {…}
@OneToOne() @MapsId @JoinColumn(name="id")
    public Student getStudent() {…}
```

- Student

```
@OneToOne(mappedBy="student") public Thesis getThesis() {…}
```

1-1 bidirectioneel: foreign key

- Een van beide entiteiten heeft een foreign key (met unique constraint)



STUDENTEN		1	0..1	MASTERPROEF	
ID	PK			ID	PK
NAAM				STUDENT	FK
…				TITEL	

- Masterproef

```
@OneToOne(optional=false)
@JoinColumn(name="student", unique=true, nullable=false,
updatable=false)
public Student getStudent() {…}
```

- Student

```
@OneToOne(mappedBy="student") public Masterproef getMasterproef() {…}
```

Relaties en cascade

UNIVERSITEIT GENT

- Actie op entiteit → actie op entiteit in relatie?
  - Bv. bestaan entiteit afhankelijk bestaan andere entiteit
- Enum javax.persistence.CascadeType
- Attribuut cascade
  - ALL
  - DETACH
  - MERGE
  - PERSIST
  - REFRESH
  - REMOVE

Customer WEG → orders WEG

```
@OneToMany(cascade=CascadeType.REMOVE, mappedBy="customer")
public Set<CustomerOrder> getOrders() { return orders; }
```