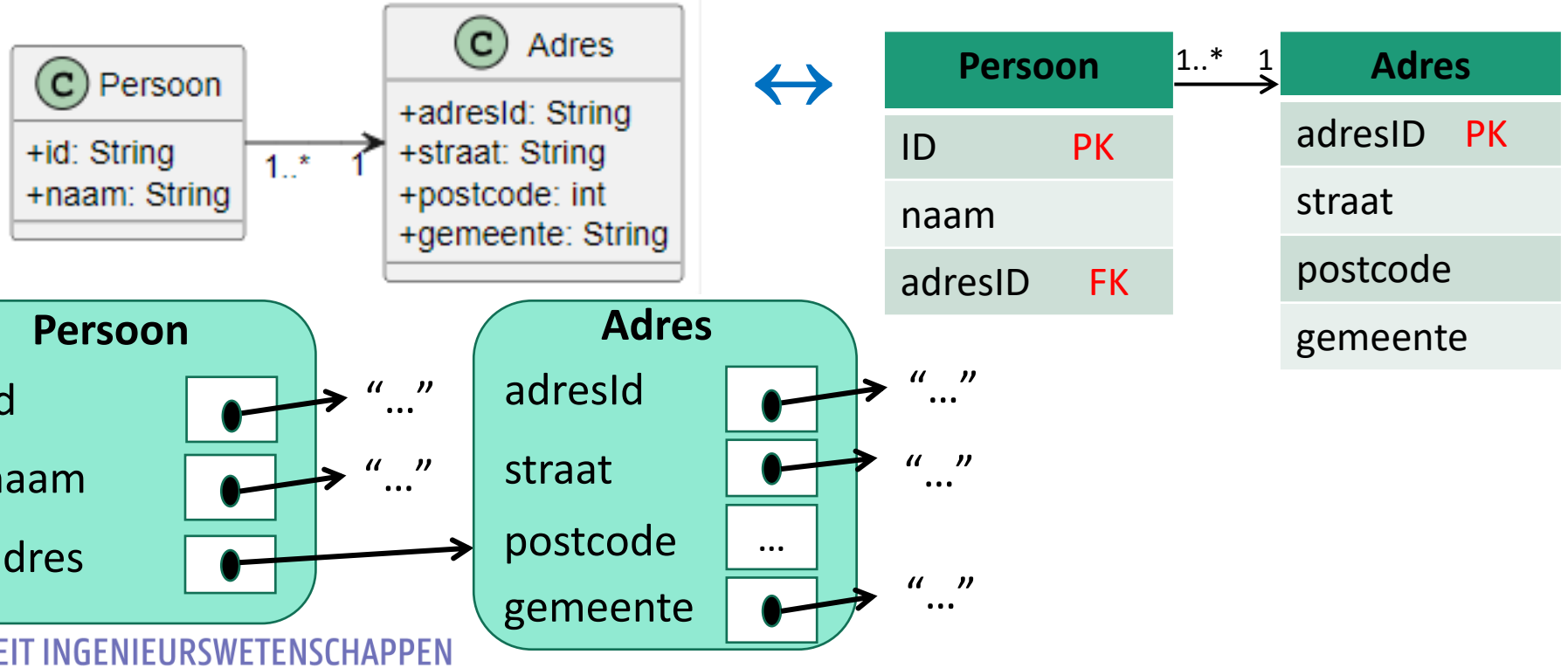
ORM associaties: value-objecten

Veerle Ongenae

Verschil OO-concepten en relationele databanken: associaties

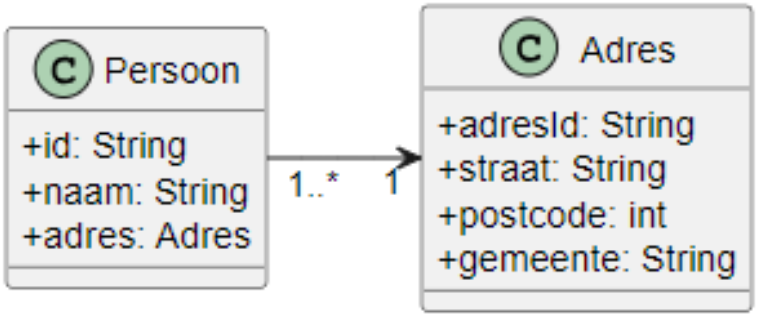
- OO: Referenties naar objecten
- DB: Foreign key



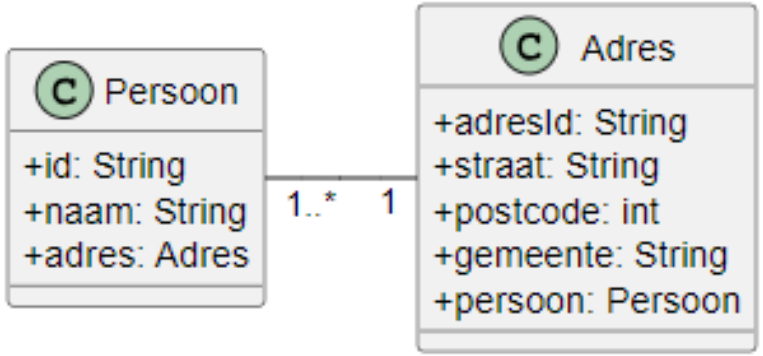
Associaties: richting

- Richting (OO, niet DB)
  - Unidirectioneel
  - Bidirectioneel
    - ➢ Moeilijk om actueel te houden
- Multipliciteit
  - 1-op-1
  - 1-op-veel
  - Veel-op-veel
- Hoe vertalen?

unidirectioneel



bidirectioneel



Opvragen geassocieerde gegevens

```
FacturatieGegevens fg = persoon.getFacturatieGegevens();
```

- Versus

```
SELECT nummer FROM facturatiegegevens f JOIN gebruikers g ON f.user_id =
g.user_id WHERE g.naam='gebruiker';
```

- Facturatiegegevens van *n* personen
  - Eerst personen opvragen, dan facturatiegegevens
    - ➢ (*1+n*) selects
  - Persoonsgegevens en facturatiegegevens tegelijk opvragen
    - ➢ 1 select

Relaties

- Value-objecten
- Relaties tussen entiteiten

Entiteiten en value-objecten

UNIVERSITEIT GENT

- Entiteit
  - Apart item in domeinmodel
  - Worden bewaard in de databank
    - ➢ Mapping vastleggen
  - Bv. Gebruiker, Factuur, …
- Attributen van een entiteit
  - Andere entiteiten
  - Value-objecten
    - ➢ Deel van een entiteit (compositie)
    - ➢ Betekenisloos indien entiteit verwijderd is
    - ➢ Bv. emailadres van een persoon

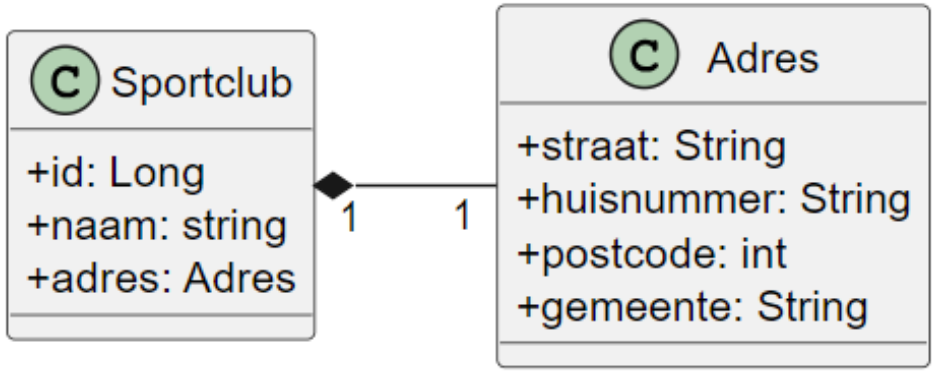
Entiteiten en value-objecten

UNIVERSITEIT GENT

- Criteria
  - Value-object hoort bij **exact één** entiteit
  - Entiteit verdwijnt → value-object verdwijnt
  - Een entiteit heeft een **identifier**
    - Identificatie om entiteit op te vragen
    - Value-object wordt opgevraagd via entiteit
- UML: compositie

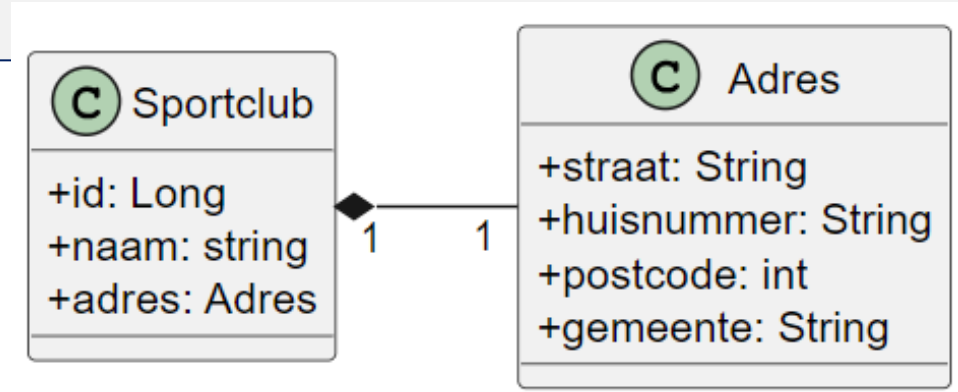
Value-objecten (embedded)

- Instantievariabele primitief type of String → één kolom in tabel (~klasse)
- Value-object niet zinvol om te bewaren in aparte tabel → meerdere kolommen in tabel
  - Klasse value-object: `@Embeddable`



SPORTCLUBS	
ID	PK
NAAM	
STRAAT	
HUISNUMMER	
POSTCODE	
GEMEENTE	

```java
@Entity
@Table(name = "sportclubs")
public class Sportclub {
    private Long id;
    private String naam;
    private Adres adres;
    …
    //@Embedded
    public Adres getAdres() {return adres;}
    public void setAdres(Adres adres) {
        this.adres = adres;}
   … }
```




```java
@Embeddable
public class Adres {
    private String straat;
    private String huisnummer;
    private Integer postcode;
    private String gemeente;

    public String getGemeente() {
        return gemeente;
    }
    …
}
```

SPORTCLUBS	
ID	PK
NAAM	
STRAAT	
HUISNUMMER	
POSTCODE	
GEMEENTE	

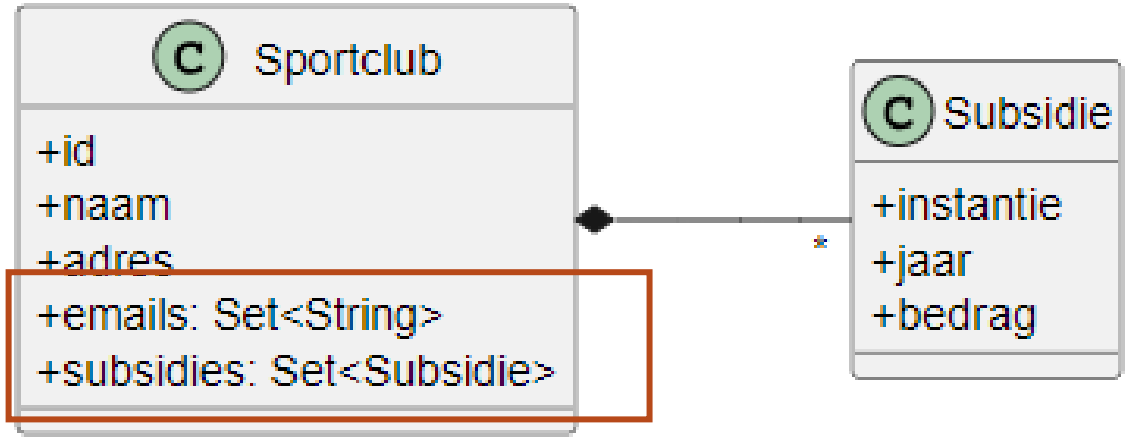
Eigenschappen zonder annotaties

Type eigenschap	Default annotatie
Primitief type of String	`@Basic` toegepast
Klasse met annotatie `@Embeddable`	`@Embedded` toegepast voor eigenschap
Serializable	`@Basic` toegepast
`java.sql.Clob` of `java.sql.Blob`	`@Lob` toegepast

Let op met collecties!

Mapping associaties value-objecten

- Instantievariabele
  - Collection primitief type of string
  - Collection ander type
- Bestaat enkel in huidige klasse/object



SPORTCLUBS	
ID	PK
NAAM	
STRAAT	
HUISNUMMER	
POSTCODE	
GEMEENTE	

EMAILADRESSEN
SPORTCLUB FK
EMAIL

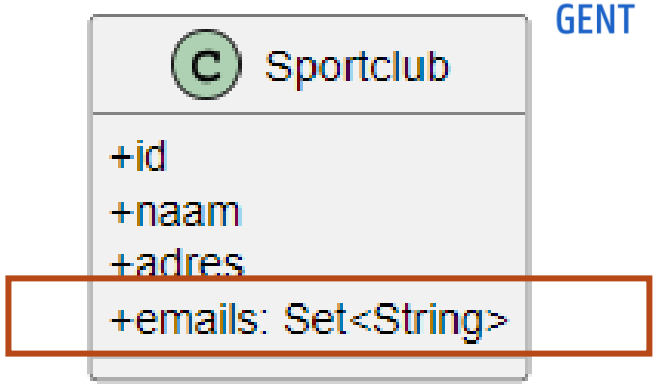
SUBSIDIES
SPORTCLUB FK
SUB_INSTANTIE
SUB_JAAR
SUB_BEDRAG

Mapping associaties value-objecten

UNIVERSITEIT GENT

- Javatype
  - Collection
  - Set
  - List
  - Map
- Type item in collectie
  - Primitief
  - Embeddable
- Annotaties
  - JPA 2.0
    - `@ElementCollection`
    - `@CollectionTable`
    - `@Column`
    - `@AttributeOverrides`

```java
public class Sportclub {
    private int id;
    private String naam;
    private Adres adres;
    private Set<String> emails;
    @ElementCollection
    @CollectionTable(name = "emailadressen",
    joinColumns = @JoinColumn(name = "sportclub"))
    @Column(name="email")
    public Set<String> getEmails() {
        return emails;
    }
    …
}
```



EMAILADRESSEN	
SPORTCLUB	FK
EMAIL	

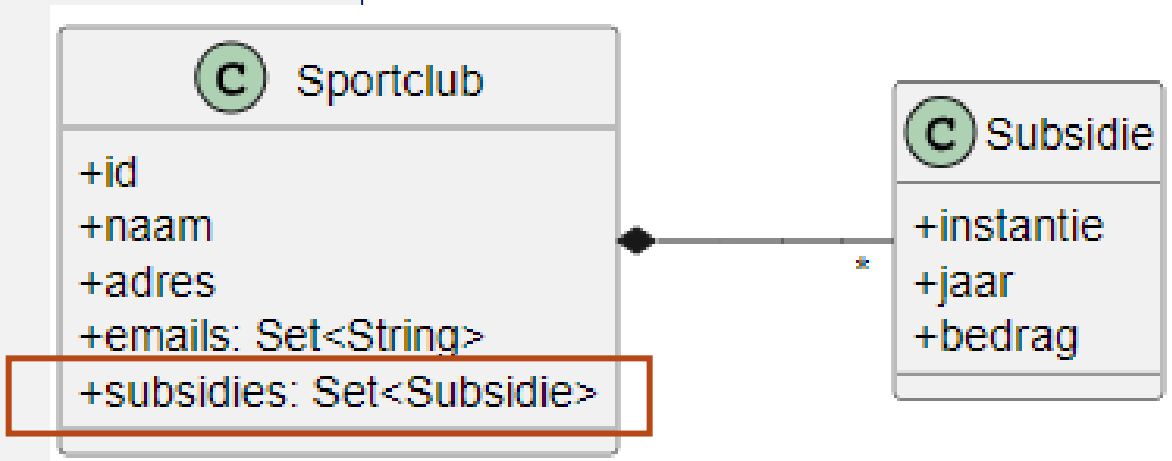
SPORTCLUBS	
ID	PK
NAAM	
STRAAT	
HUISNUMMER	
POSTCODE	
GEMEENTE	

Mapping associaties value-objecten

UNIVERSITEIT GENT

- Collection primitief type of string
  - Set, List ↔ `@ElementCollection`
  - `@CollectionTable`
    - Koppeling tussen tabel object en tabel instantievariabele
    - `name`: naam tabel
    - `joinColumns`: kolom(men) foreign key
      - `@JoinColumn`: naam kolom met foreign key
  - `@Column`
    - Mapping element uit collection
    - `name`: kolom waarde uit collection

```java
public class Sportclub {
    private int id;
    private String naam;
    private Adres adres;
    private Set<String> emails;
    private List<Subsidie> subsidies;
    @ElementCollection
    @CollectionTable(name = "subsidies",
        joinColumns = @JoinColumn(name = "sportclub"))
    @AttributeOverrides({
        @AttributeOverride(name = "jaar",
        column = @Column(name = “sub_jaar")),
        @AttributeOverride(name = "bedrag",
        column = @Column(name = “sub_bedrag")),
        @AttributeOverride(name = "instantie",
        column = @Column(name = “sub_instantie"))
    })
    public Set<Subsidie> getSubsidies() {
        return subsidies;
    }    …
```



SPORTCLUBS		SUBSIDIES
ID	PK	SPORTCLUB FK
NAAM		SUB_INSTANTIE
STRAAT		SUB_JAAR
HUISNUMMER		SUB_BEDRAG
POSTCODE		
GEMEENTE		

Mapping associaties value-objecten

UNIVERSITEIT GENT

- Collection geen primitief type of string
  - Set, List ↔ `@ElementCollection`
  - `@CollectionTable`
    - Koppeling tussen tabel object en tabel instantievariabele
    - `name`: naam tabel
    - `joinColumns`: kolom(men) foreign key
      - `@JoinColumn`: naam kolom met foreign key
  - `@AttributesOverrides`
    - Mapping eigenschappen element uit collection op kolommen
    - `@AttributeOverride`
      - Eigenschap element ↔ kolom
      - `name`: eigenschap element uit collection
      - `column`: corresponderende kolom in tabel