UML Class Diagrams (Relationships)

Software Design (40007) – 2023/2024

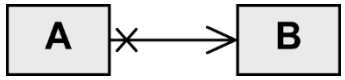
Justus Bogner
Ivano Malavolta

Software and Sustainability research group (S2)
Department of Computer Science, Faculty of Sciences

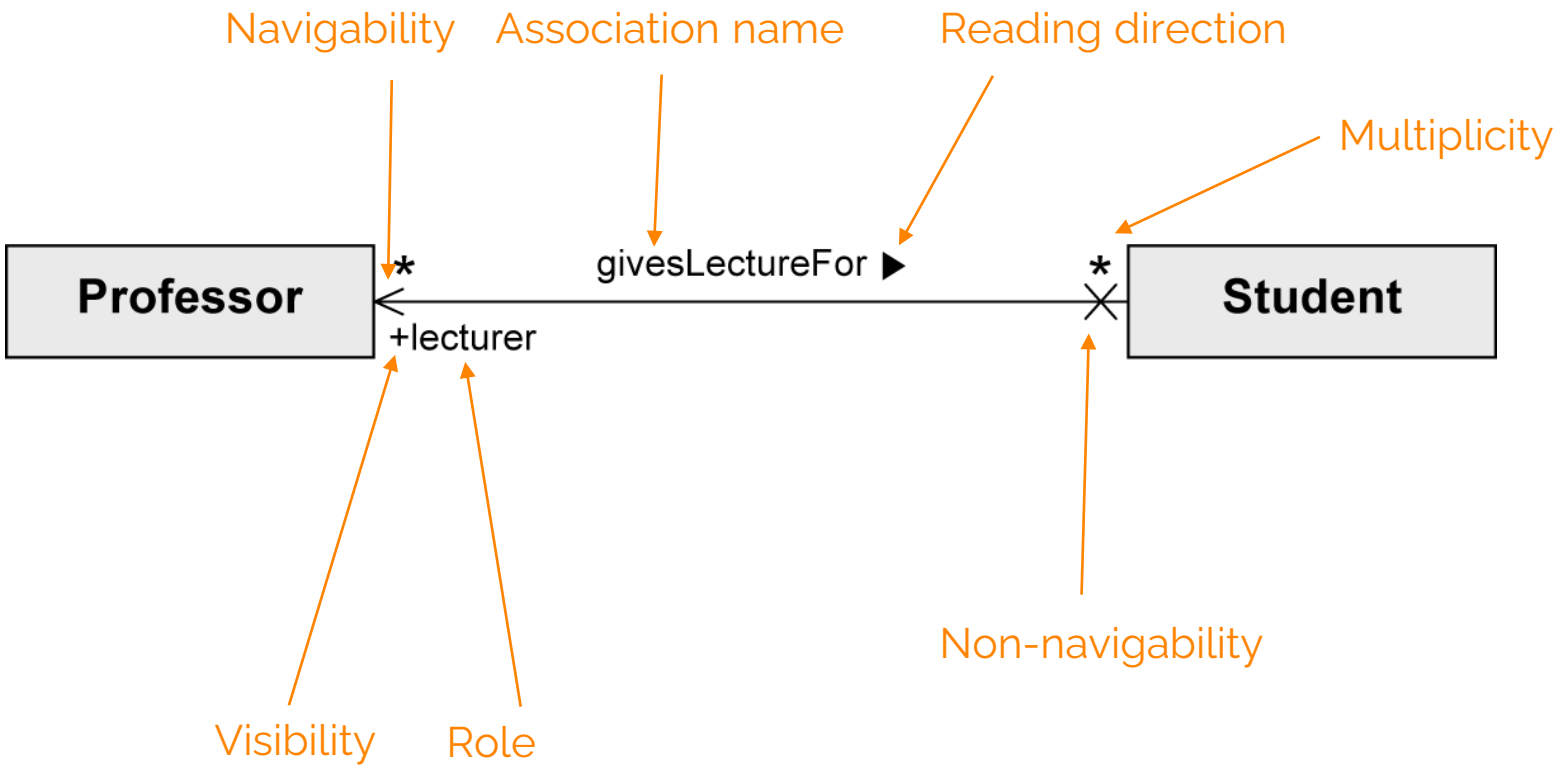
Roadmap

- Relationships
  - Binary Association
  - N-ary Association
  - Association Class
  - Aggregation / Composition
  - Generalization (inheritance and abstract classes)

Binary association



Connects instances of two classes with one another



Binary association - navigability

Navigability: an object knows its partner objects and can therefore access their visible attributes and operations

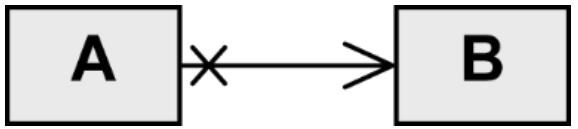
- Indicated by open arrow head

Non-navigability

- Indicated by cross, but this is also often left out

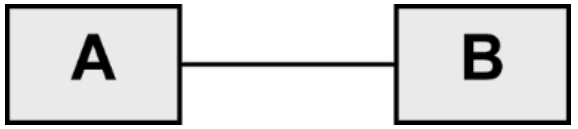
Example:

- **A** can access the visible attributes and operations of **B**
- **B** cannot access any attributes and operations of **A**



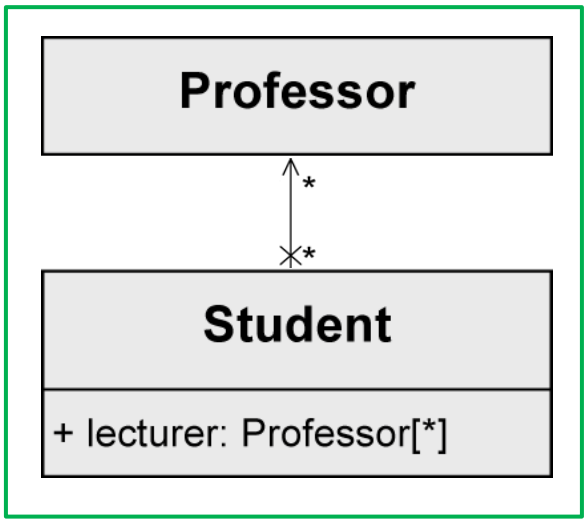
Navigability undefined

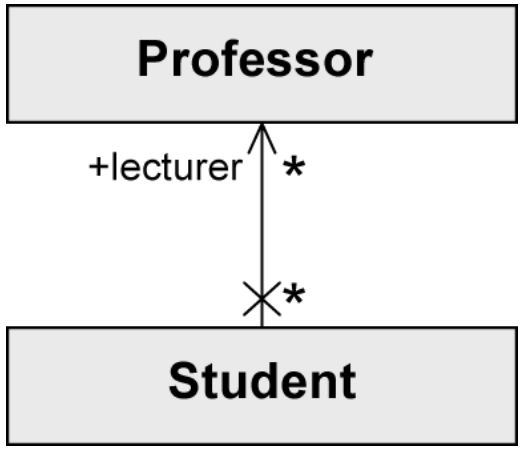
- Bidirectional navigability is assumed



Binary association representation

Preferable





In Java:

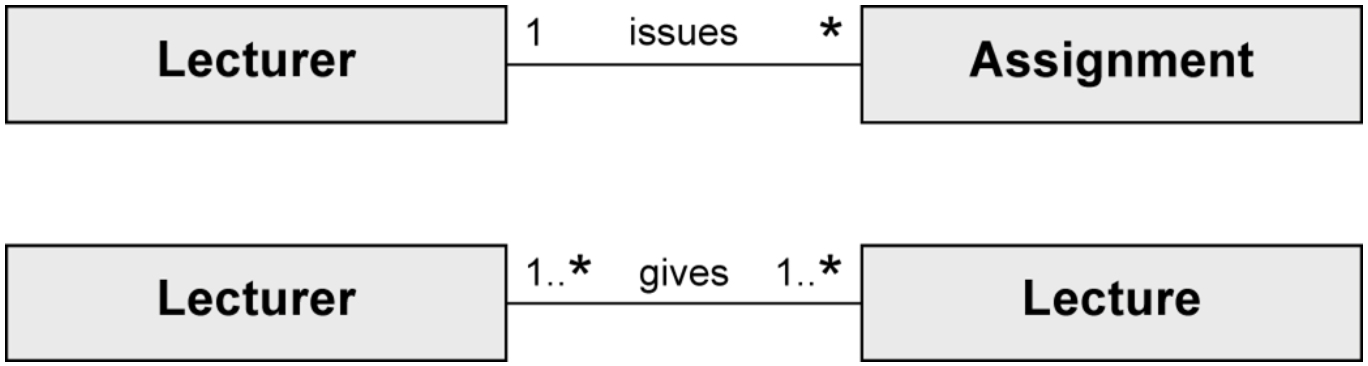
```
class Professor {…}

class Student {
  public Professor[] lecturer;
  …
}
```

See anything here that is not ideal?

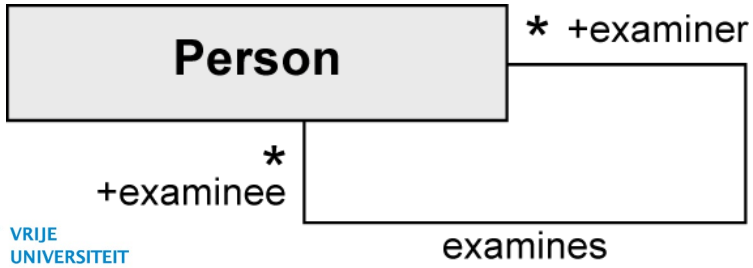
A collection should be reflected in the variable name, e.g., `lecturers` or `lecturerList`.

Binary association – multiplicity and role

Multiplicity: Number of objects that may be associated with exactly one object of the opposite side

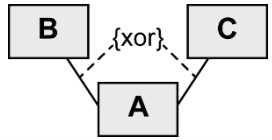


Role: describes the way in which an object is involved in an association relationship

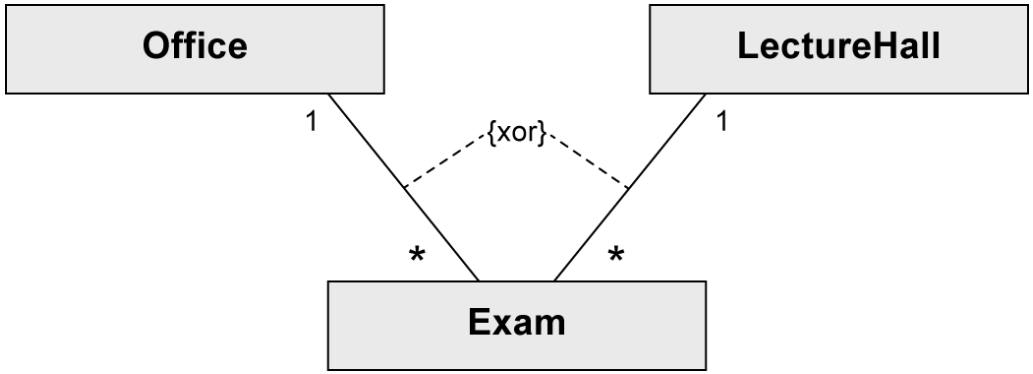



```java
public class Person {
    public ArrayList<Person> examiner;
    public ArrayList<Person> examinee;
}
```

Binary association – xor constraint

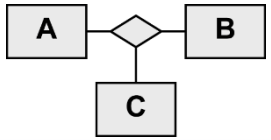


"exclusive or" constraint:
An object of class `Exam` is associated with an object of class `Office` **or** an object of class `LectureHall`, **but not with both**

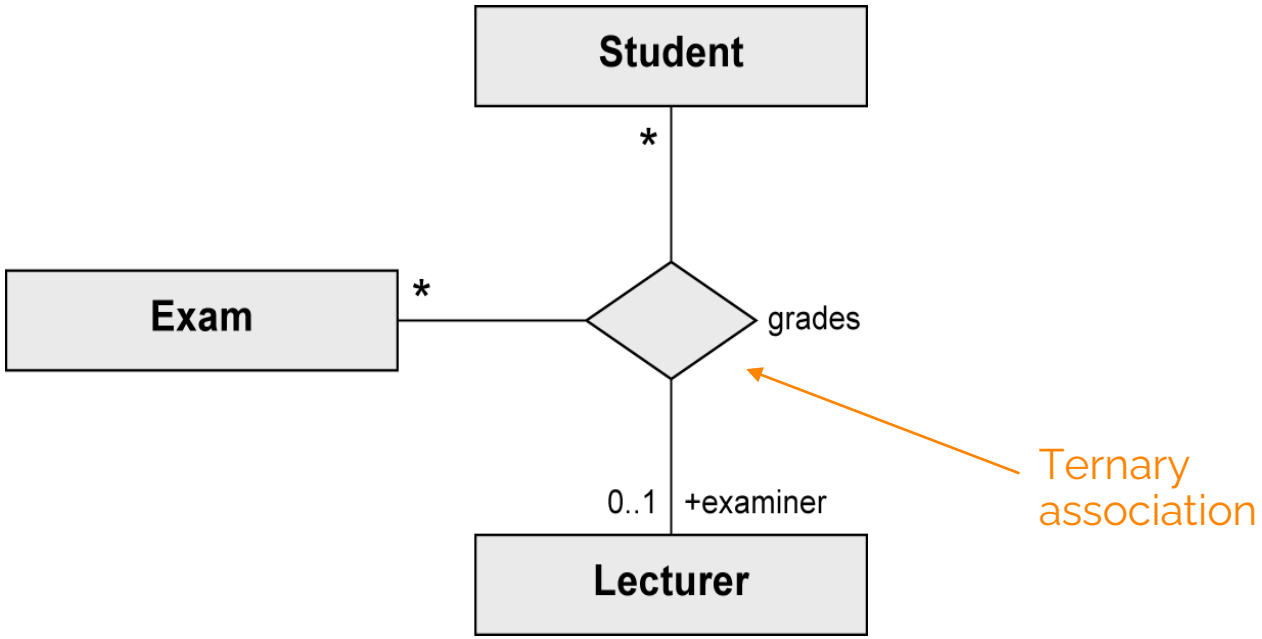


How would you implement it in Java?

n-ary association (1/2)



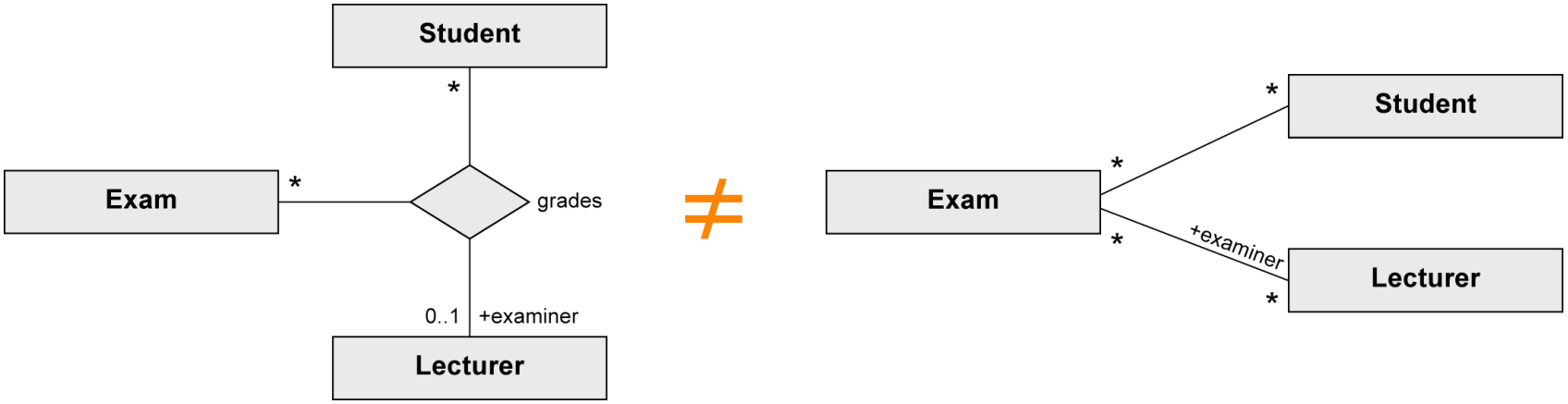
- More than two objects are involved in the relationship
- No navigation directions



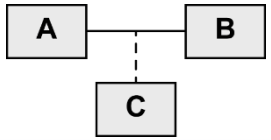
n-ary association (2/2)

Example

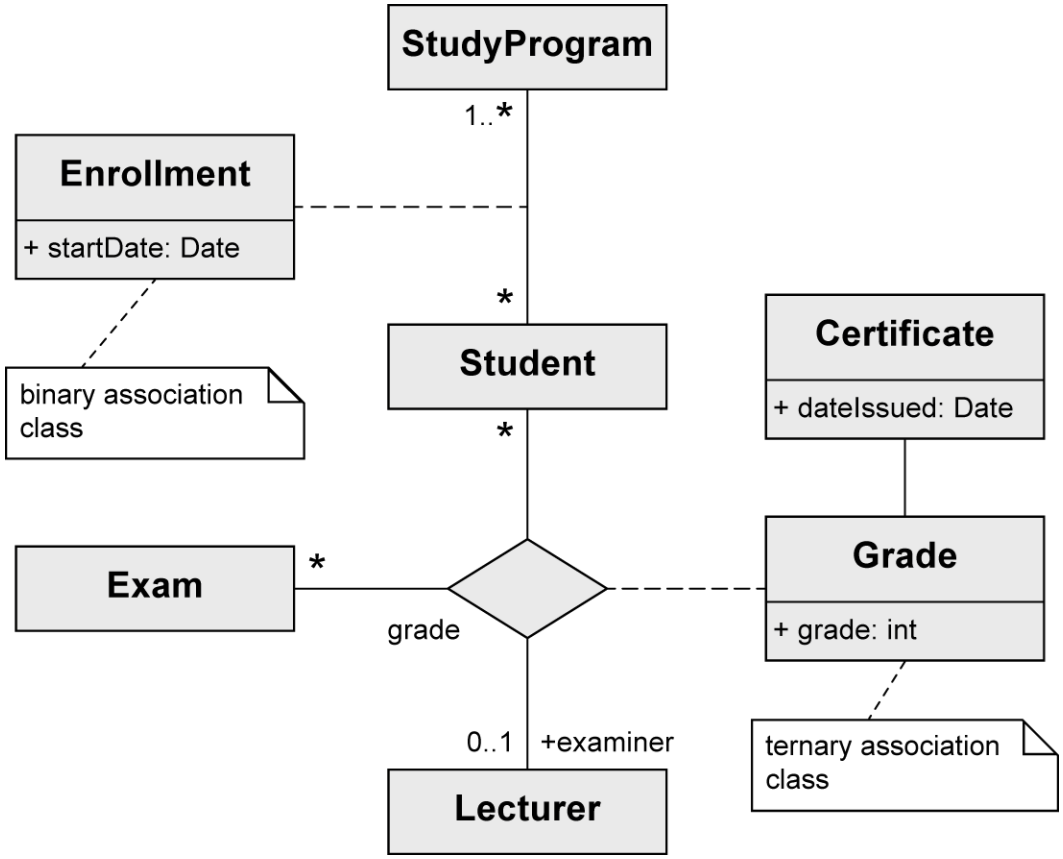
- (**Student**, **Exam**) → (**Lecturer**)
  - One student takes one exam with either one or no lecturer
- (**Exam**, **Lecturer**) → (**Student**)
  - One exam with one lecturer can be taken by any number of students
- (**Student**, **Lecturer**) → (**Exam**)
  - One student can be graded by one lecturer for any number of exams



Association class

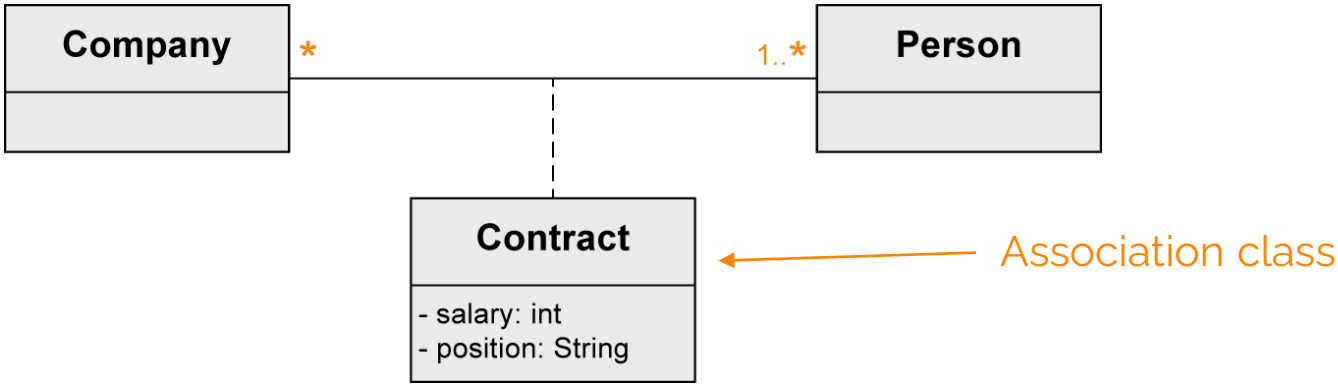


Assign attributes to the relationship between classes rather than to a class itself

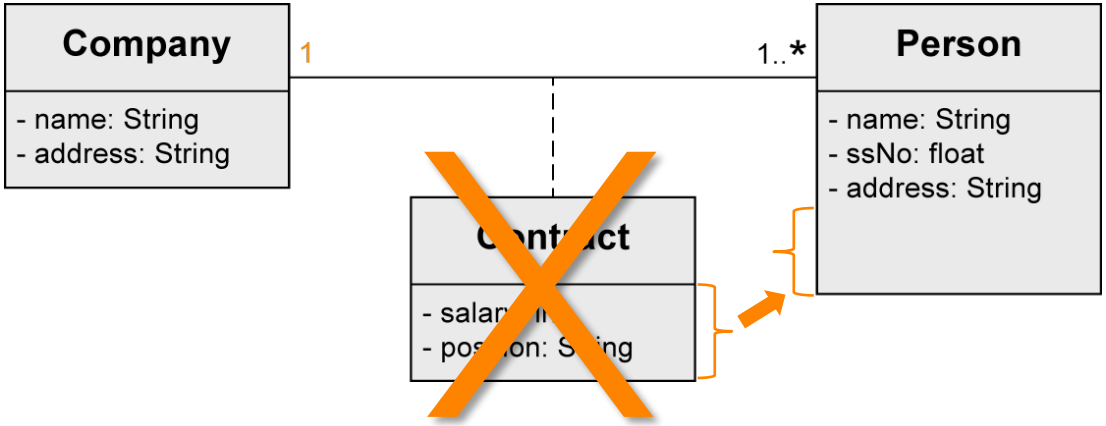


When to use an association class

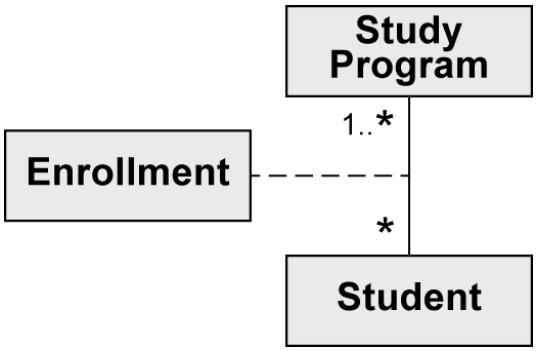
- Mandatory when modeling n:m associations



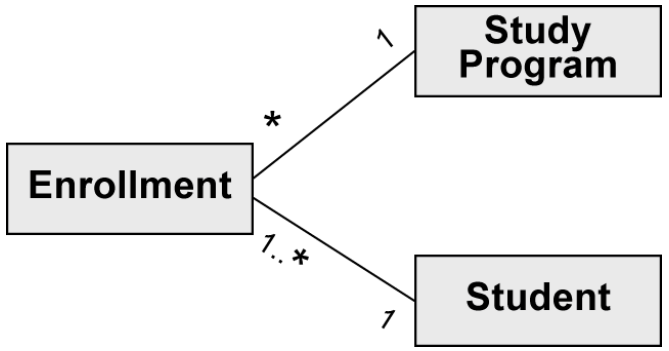
- With 1:1 or 1:n → possible but not mandatory



Association class vs. regular class



≠



A `Student` can enroll for one particular `StudyProgram` only once

A `Student` can have multiple `Enrollments` for one and the same `StudyProgram`

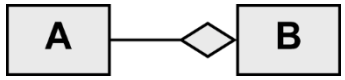
Aggregation

- Special form of association
- Used to express that a class **is part of** another class

- Properties of the aggregation association:
  - **Transitive**: if **B** is part of **A** and **C** is part of **B**, **C** is also part of **A**
  - **Asymmetric**: it is not possible for **A** to be part of **B** and **B** to be part of **A** simultaneously

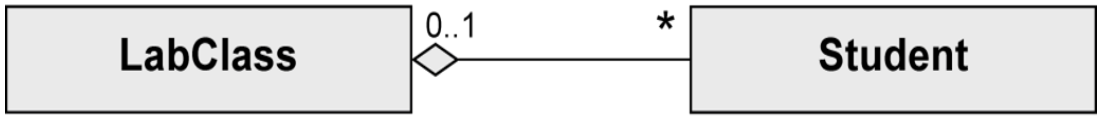
- Two types:
  - (Shared) aggregation
  - Composition

(Shared) Aggregation

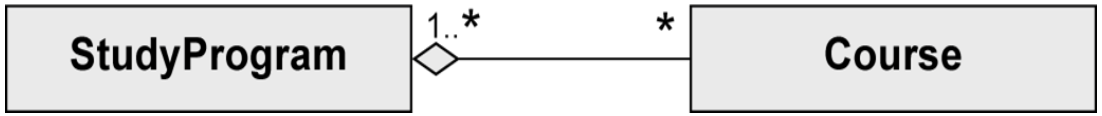


- Expresses a weak belonging of the parts to a whole
  - Parts also exist independently of the whole
- Multiplicity at the aggregating end may be >1
  - One element can be part of multiple other elements simultaneously

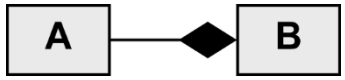
- Example:
  - **Student** is part of **LabClass**



  - **Course** is part of **StudyProgram**



Composition

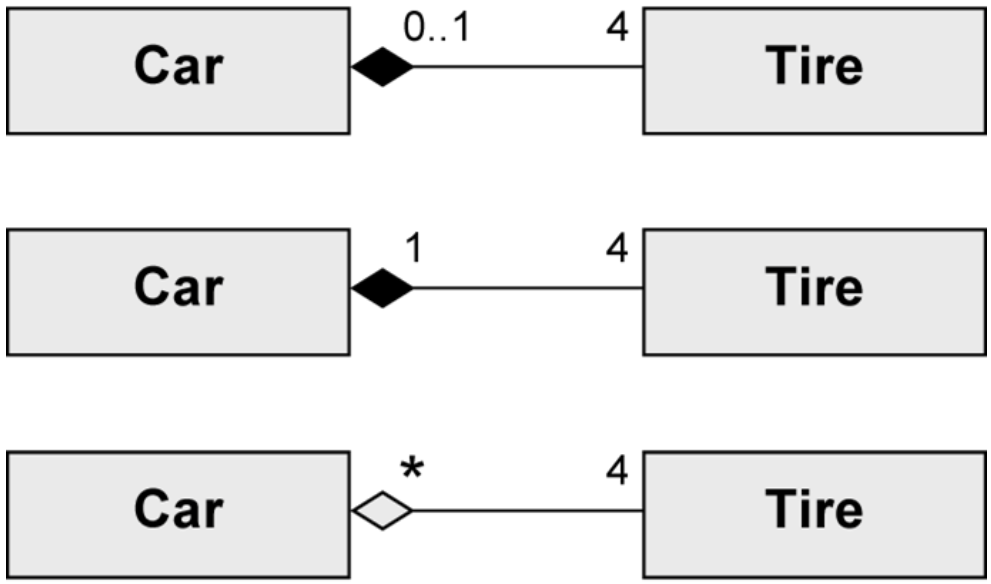


- **Existence dependency** between the composite object and its parts
- An existing part is contained in at most one composite object at one specific point in time
  - Multiplicity at the aggregating end max 1

- If the composite object is deleted, its parts are also deleted
  → the part usually cannot exist without the whole
  (exceptions are possible with multiplicity `0..1` instead of `1`)

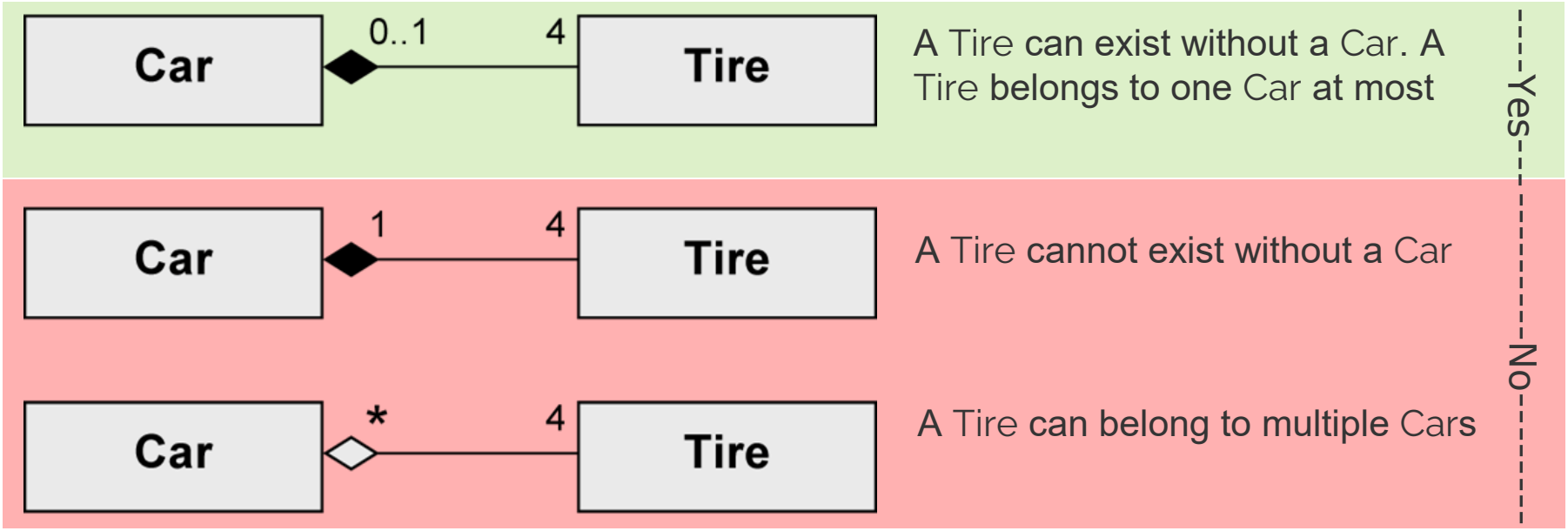
Aggregation vs. composition

Which one is a valid representation of reality?

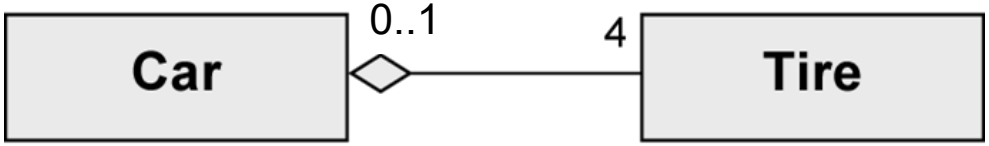


Aggregation vs. composition

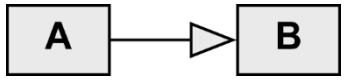
Which one is a valid representation of reality?



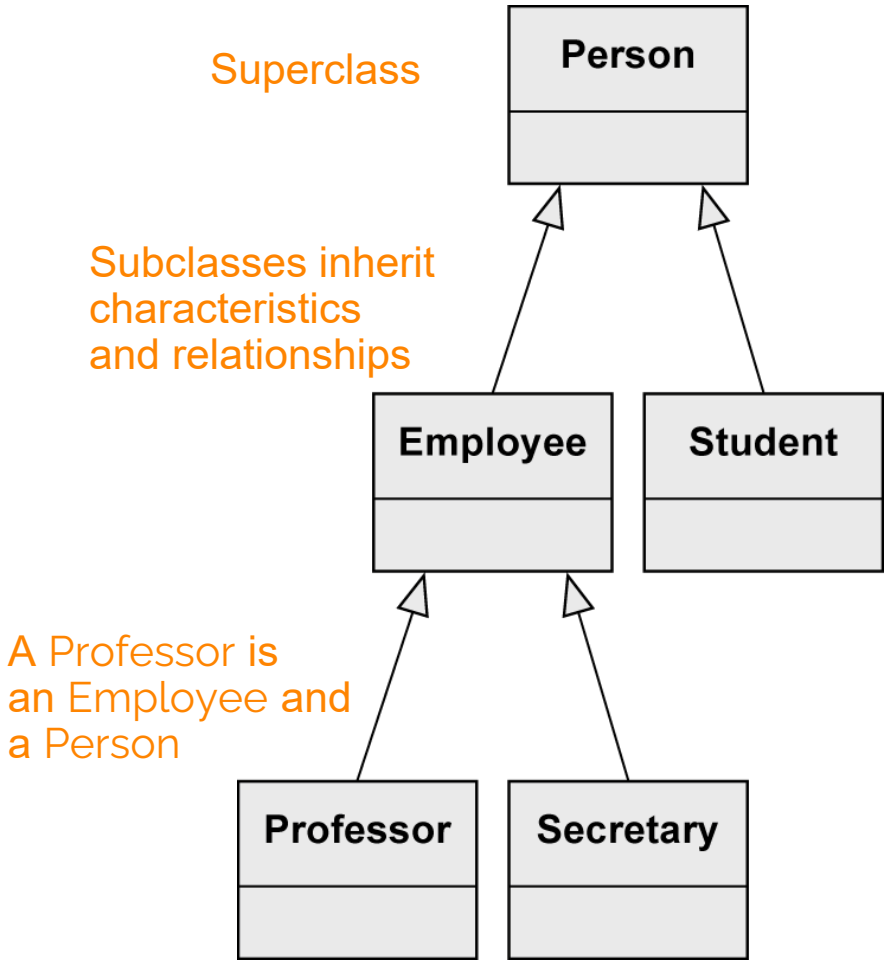
Another valid option:



Generalization - inheritance



- Attributes, operations, and relationships of the general class **are passed on to its subclasses** (except private ones)
- Every instance of a subclass is also an indirect instance of the superclass
- Subclasses may have further characteristics and relationships
- Generalizations are transitive



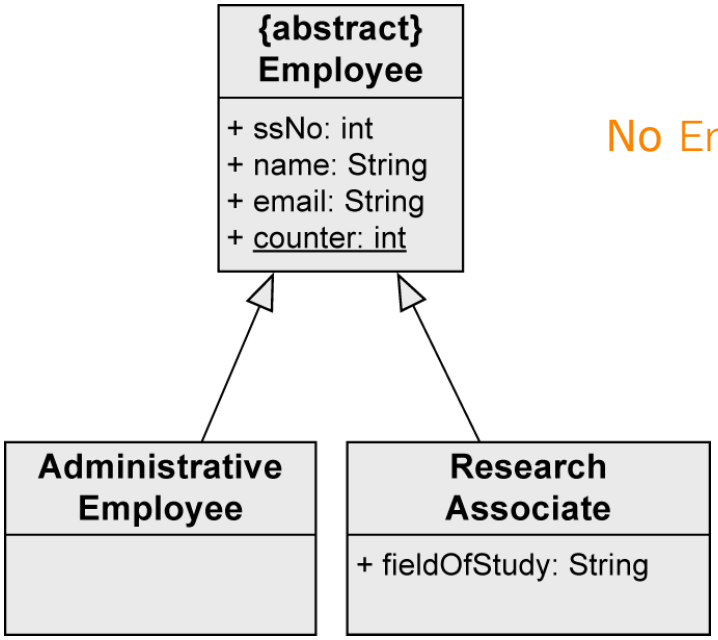
Generalization – abstract classes

{abstract}
A

- Used to highlight common characteristics of their subclasses while ensuring **no direct instances of the superclass**
- Only its non-abstract subclasses can be instantiated
- Useful in the context of generalization relationships

{abstract}
Person

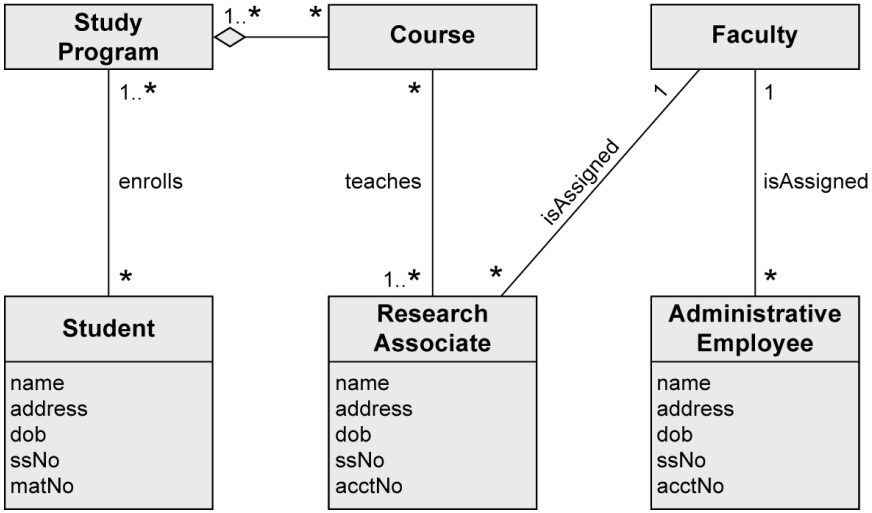
Person

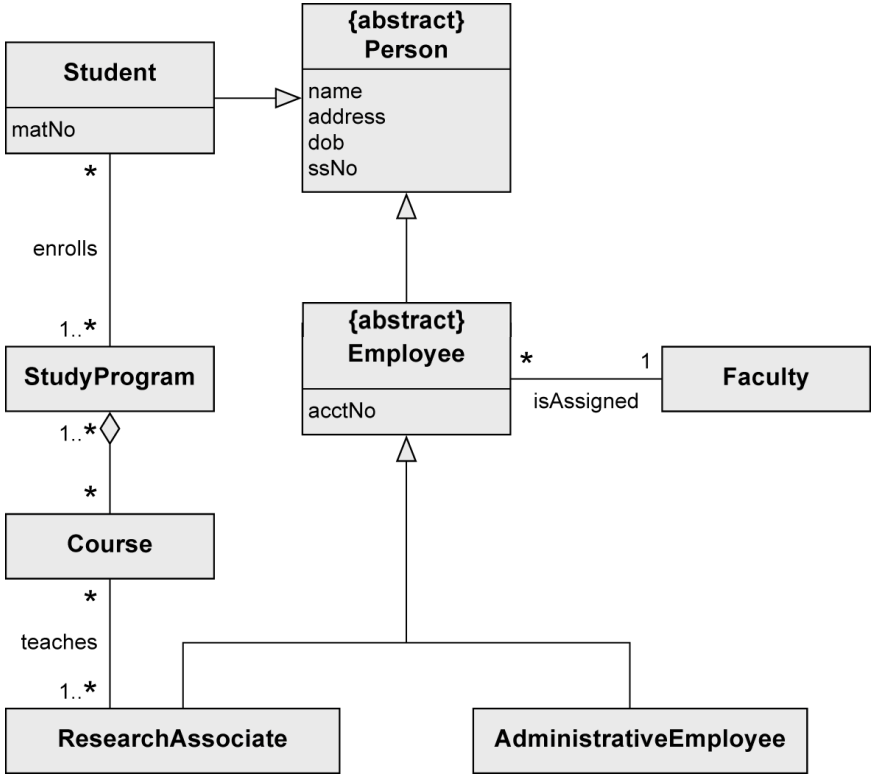


No Employee object possible

Two types of Employees: Administrative Employee and Research Associate

Example: generalization avoids duplication





Do you see something "incorrect" here?