

# Chapter 3: UML

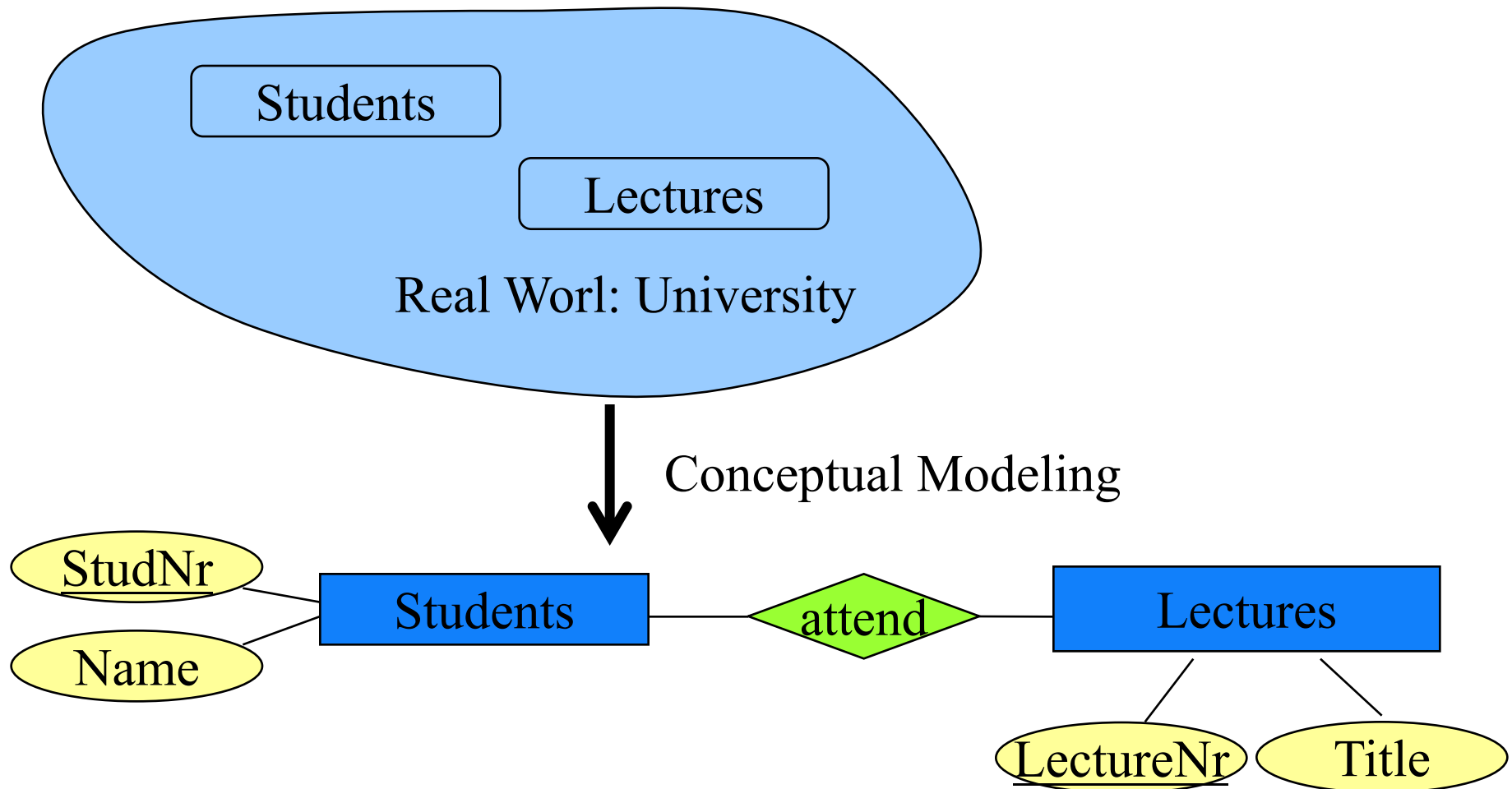
## Content:

- Learn how to draw UML diagrams
- UML is an alternative way to model a database

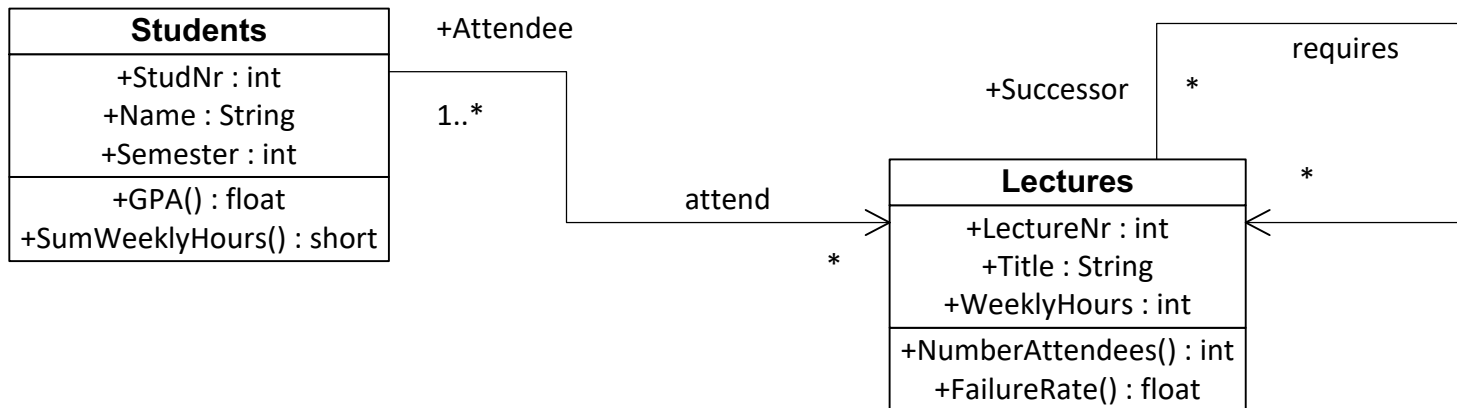
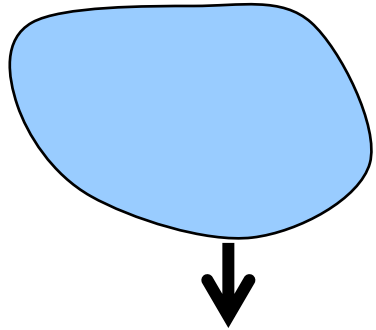
## Next:

- Convert UML and ER diagrams into a database schema

# Modeling a small example application: E/R



# Modeling a small example application: UML



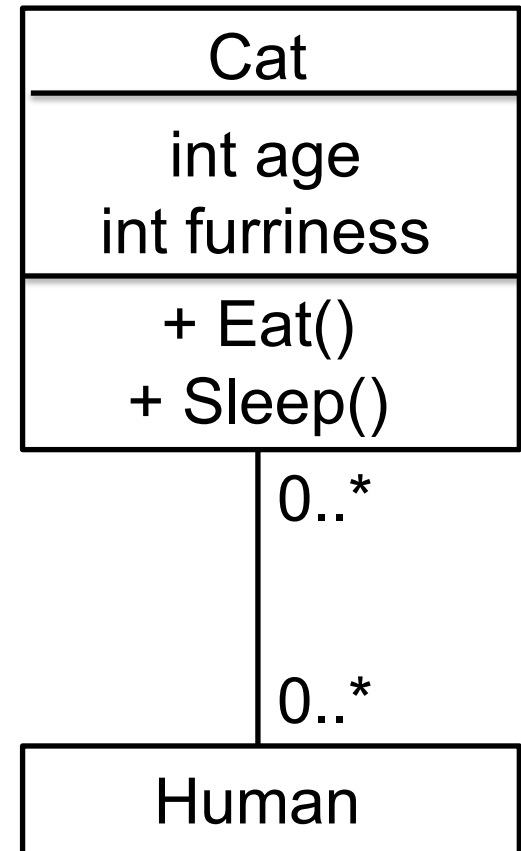
# Data modelling with UML

- UML = Unified Modelling Language
- De facto standard for object oriented software design
- Several diagrams, we focus on class diagrams

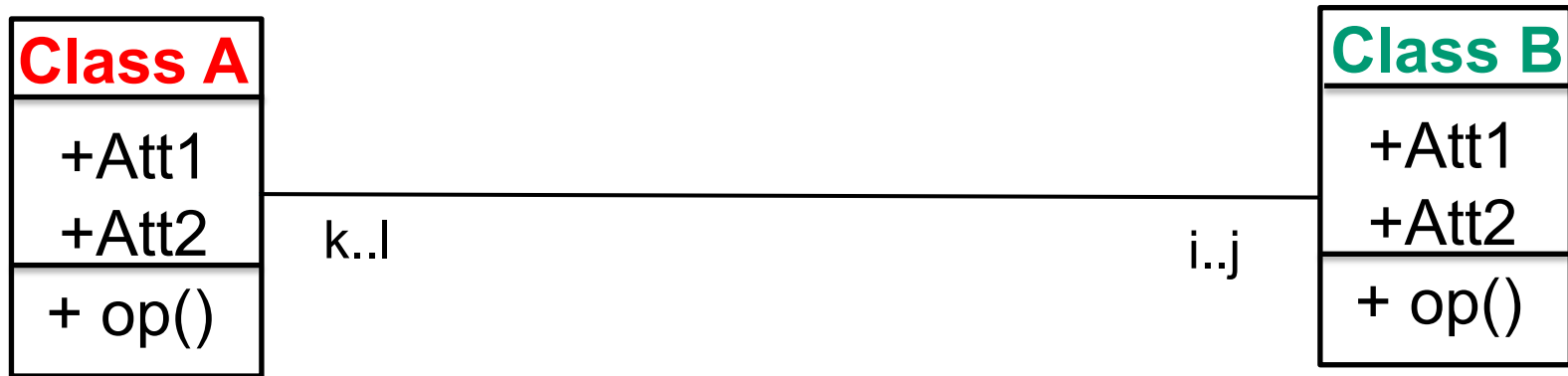
# Data modelling with UML

Main concepts in UML class diagrams:

- **Classes:** models similar objects according to:
  - Structure (~Attributes)
  - Behavior (~Operations/Methods)
  - ≈ Entities in ER-Diagram
- **Associations:** between classes correspond to relationships
  - Generalization, Aggregation, ...
  - ≈ Relationship in ER-Diagram
- **Multiplicities:** for associations
  - 0..\* to 0..\*, 1 to 1, ...
  - ≈ Functionalities in ER-Diagram



# Multiplicity



- Every element of **Class A** is associated with at least  $i$  elements of **Class B** and with at most  $j$  elements of **Class B**
- Analogously for the interval  $k..l$
- Multiplicity is analogous to the functionalities in the ER-Model not to the (min, max)-Notation: Watch out!

# UML Association Types

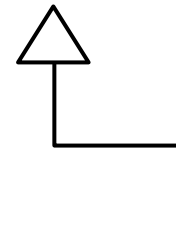
## Association:

- Generic relationship
- Any multiplicity possible



## Generalization:

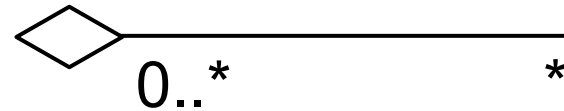
- “Is-a” relationship
- Inheritance in Java/C++



# UML Association Types

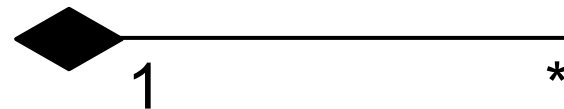
## Aggregation:

- “Part-of” or “has”
- Multiple owners



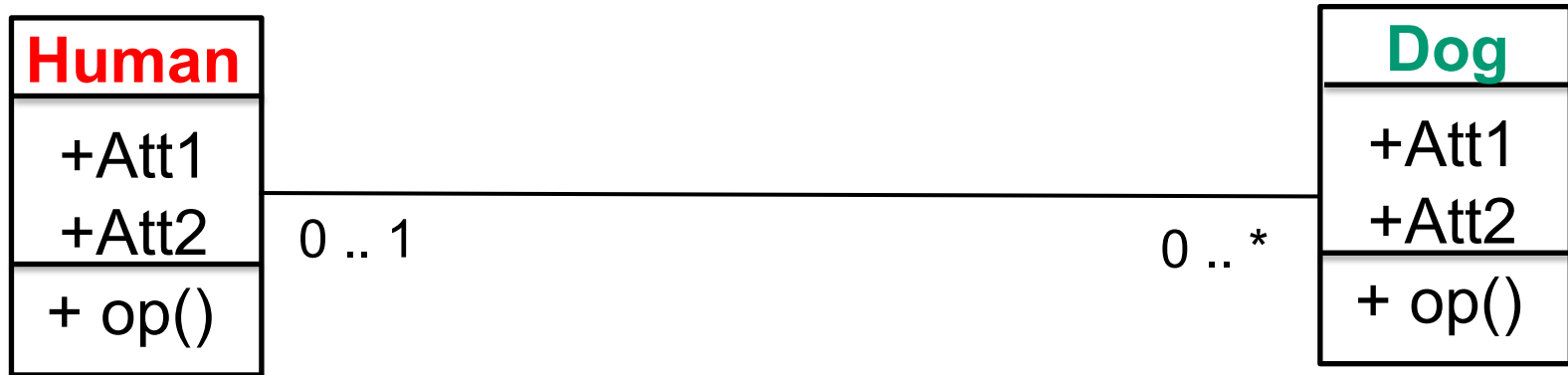
## Composition:

- Special case of Aggregation
- Existence dependent
- Exactly one owner



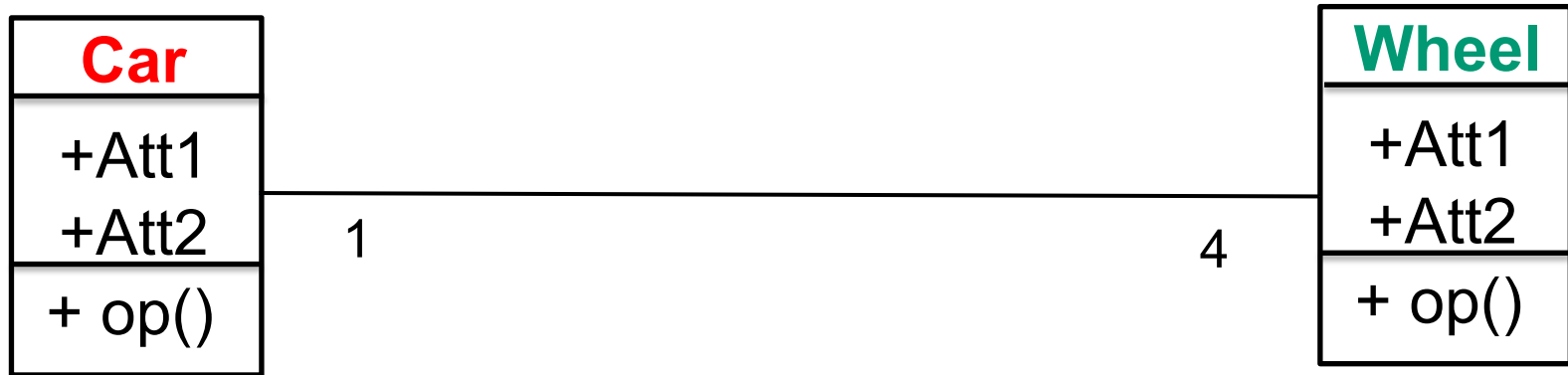


# Multiplicity: Example 1



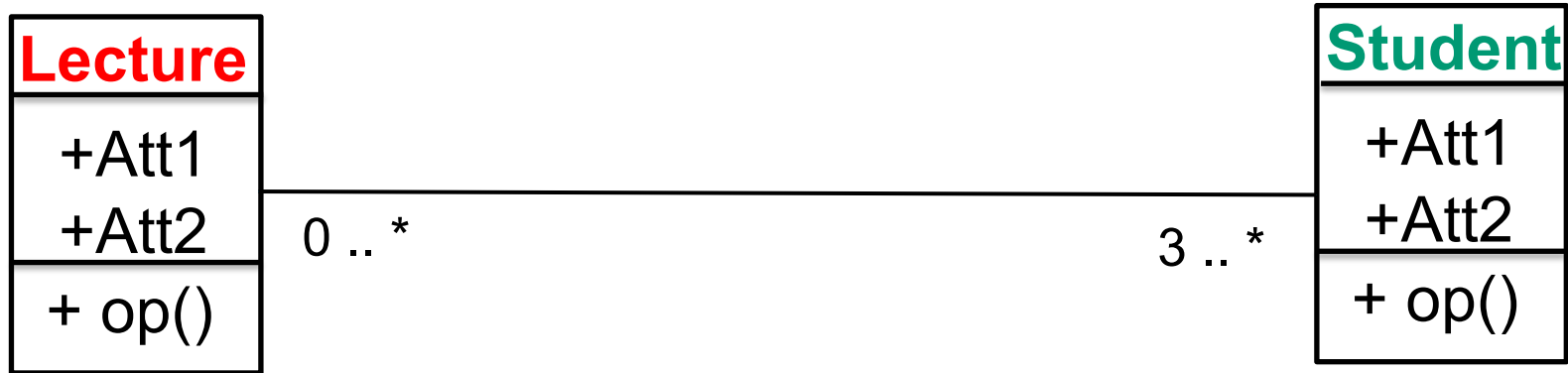
- A **Human** can have an arbitrary amount of **Dogs**
- A **Dog** might has 0 or 1 **Humans**
- Association type: Regular association or aggregation

# Multiplicity: Example 2



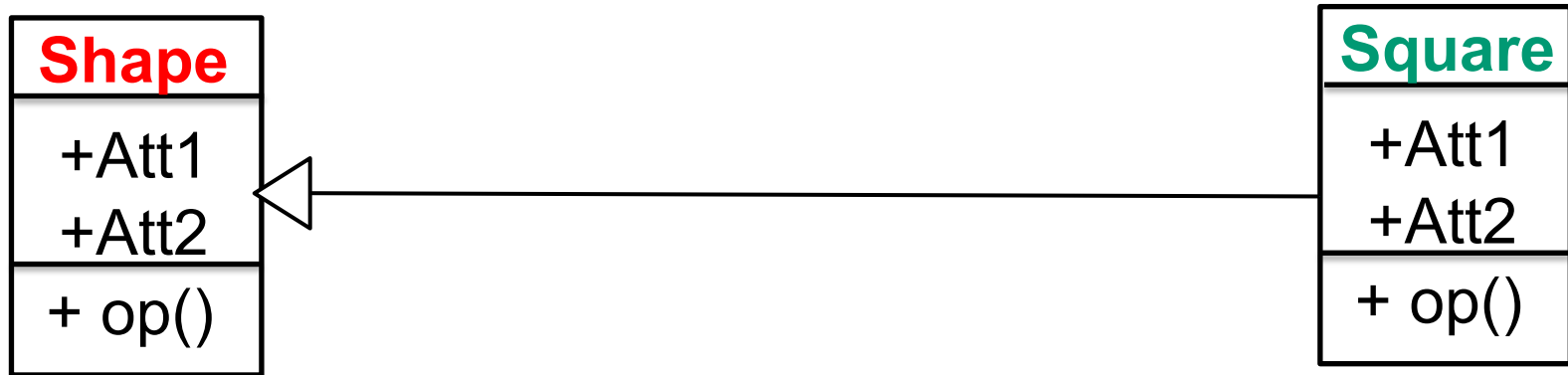
- A **Car** has exactly 4 **Wheels**
- A **Wheel** belongs to exactly 1 **Car**
- Association type: Composition or aggregation

# Multiplicity: Example 3



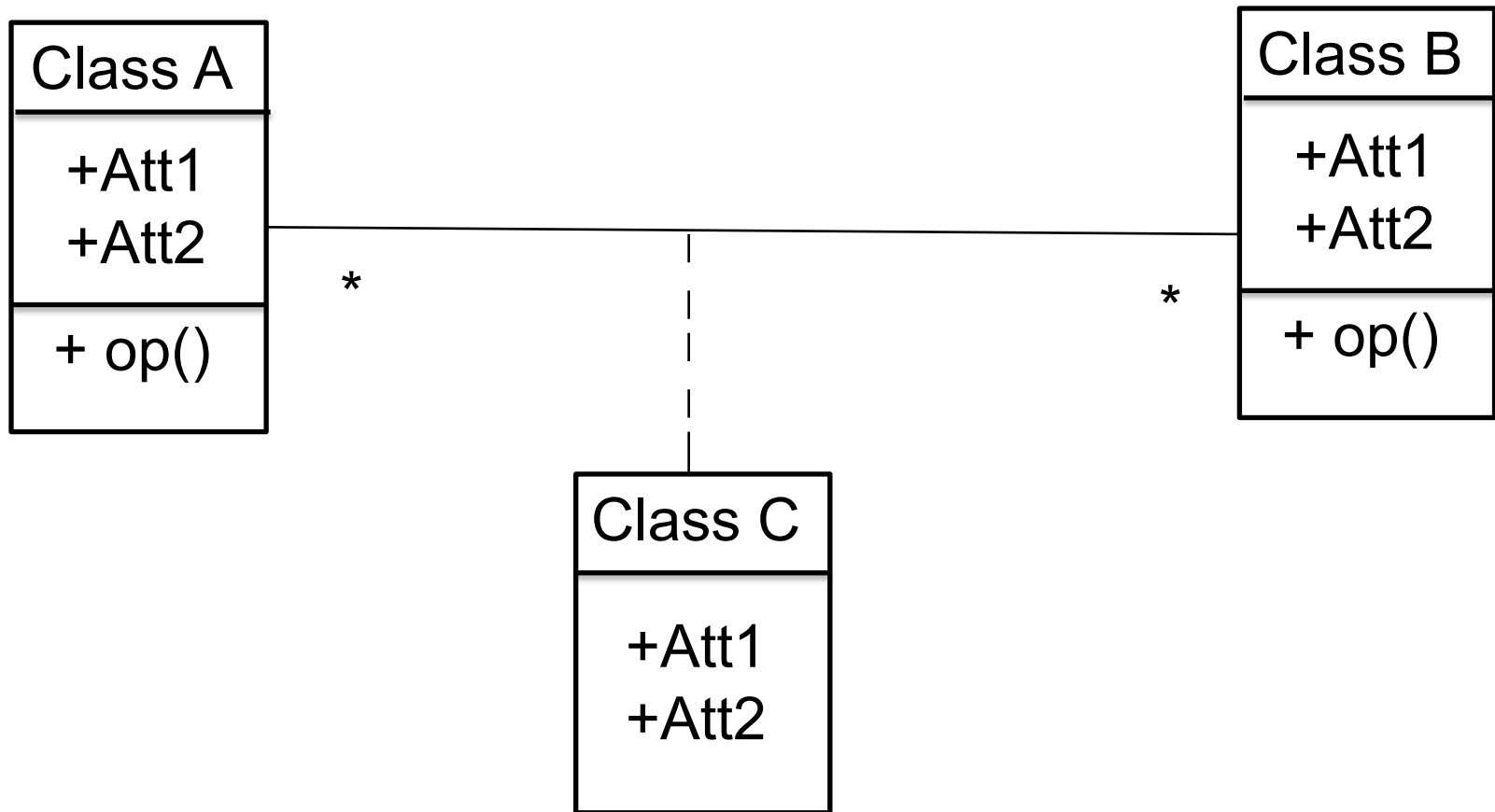
- A **Lecture** has to have at least 3 **Students**
- A **Student** can attend any number of **Lectures**
- Association type: Aggregation

# Multiplicity: Example 4



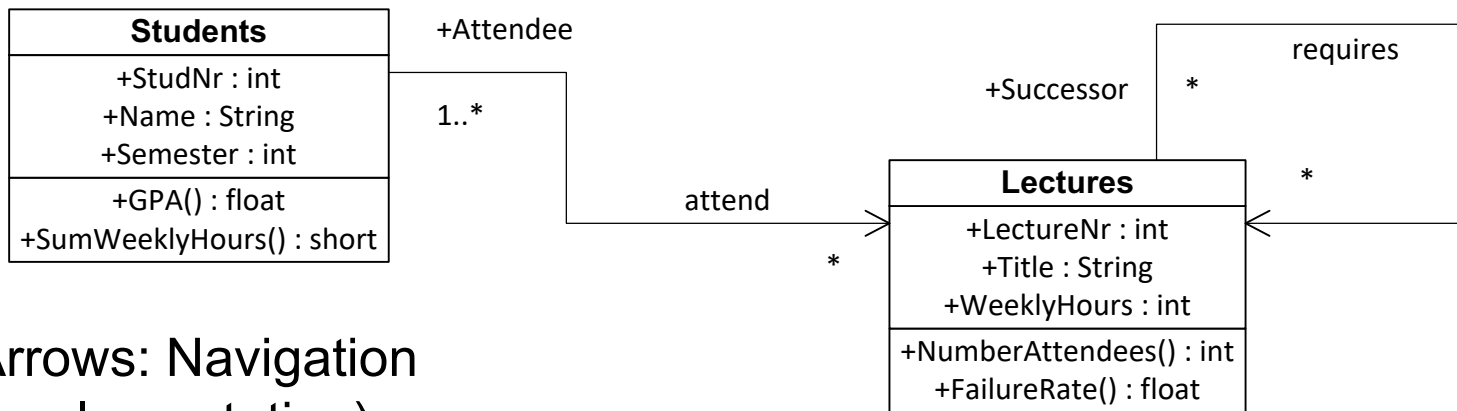
- A **Square** is a **Shape**
- Association type: Generalization

# Association class



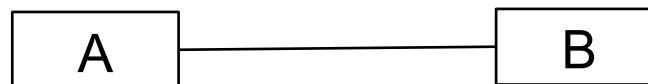
... for attributes of the association

# Navigation

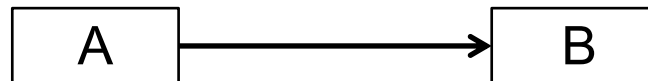


## Arrows: Navigation (Implementation)

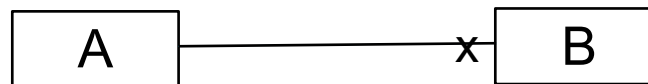
No statement on navigation



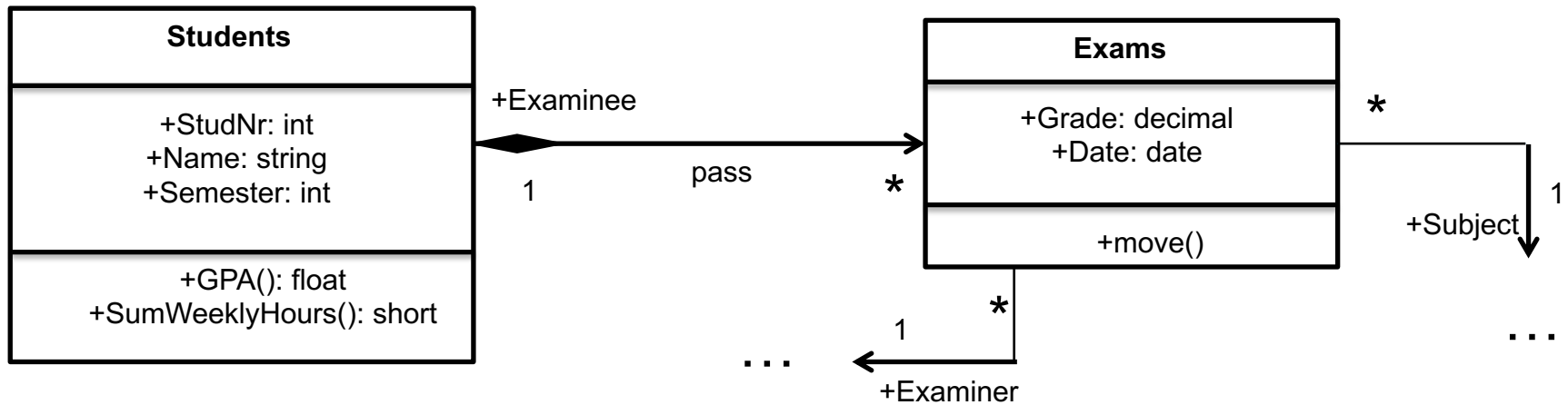
Navigation from A to B allowed



Navigation from A to B forbidden



# Composition



<b>Studenten</b>
+MatrNr : int +Name : String +Semester : int
+Notenschnitt() : float +SummeWochenstunden() : short

<b>Vorlesungen</b>
+VorlNr : int +Titel : String +SWS : int
+AnzHörer() : int +DurchfallQuote() : float

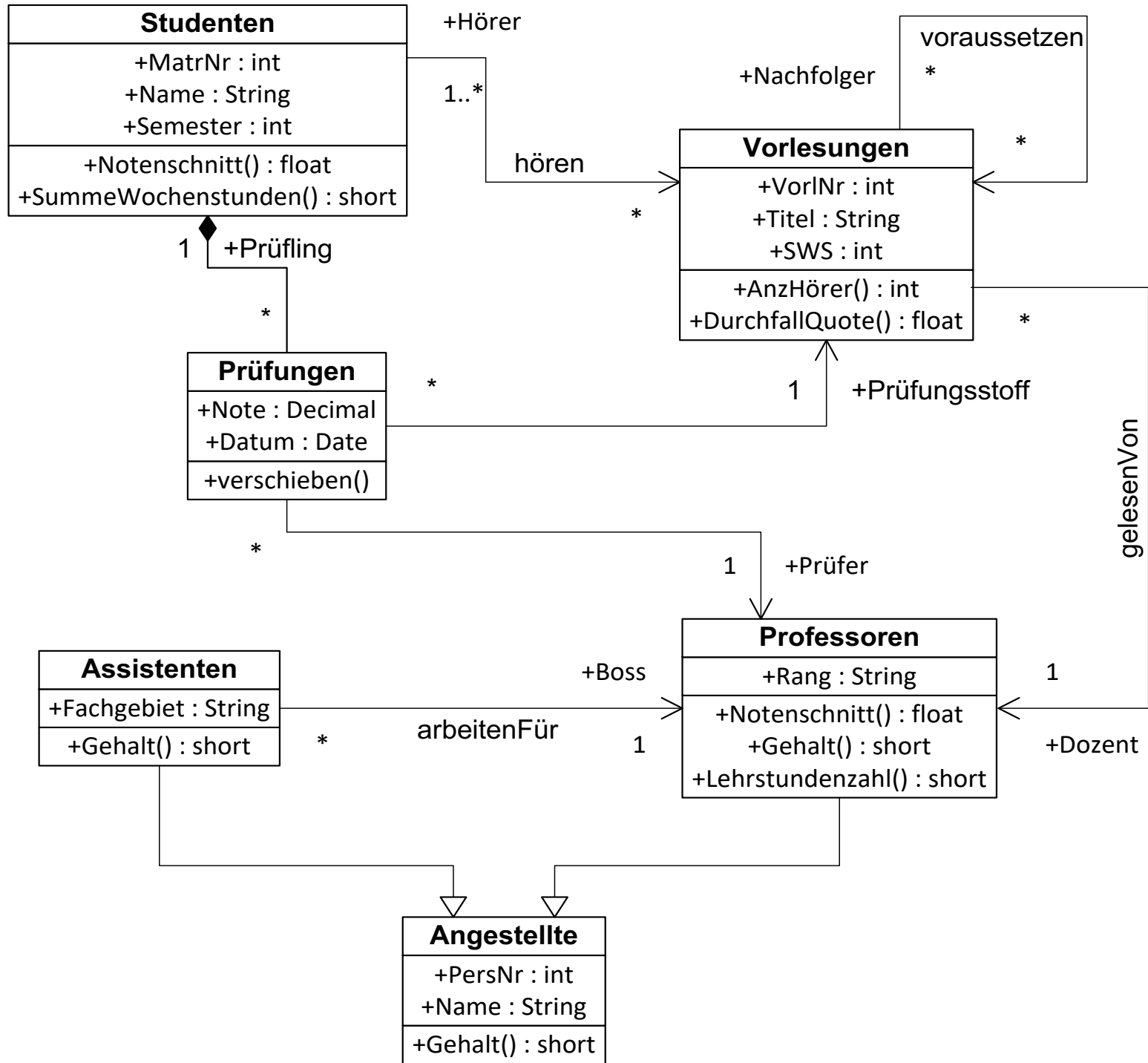
<b>Prüfungen</b>
+Note : Decimal +Datum : Date
+verschieben()

<b>Assistenten</b>
+Fachgebiet : String
+Gehalt() : short

<b>Professoren</b>
+Rang : String
+Notenschnitt() : float +Gehalt() : short +Lehrstundenzahl() : short

<b>Angestellte</b>
+PersNr : int +Name : String
+Gehalt() : short





# Other useful Diagrams

- Use Case Diagram
- Interaction Diagram
- Sequence Diagram

Search for examples in the internet

# Quiz UML

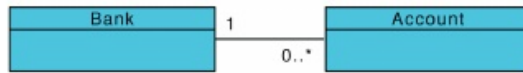
From the Stanford MOOC:

[https://lagunita.stanford.edu/courses/DB/UML/SelfPaced/courseware/ch-unified\\_modeling\\_language/seq-quiz-uml/](https://lagunita.stanford.edu/courses/DB/UML/SelfPaced/courseware/ch-unified_modeling_language/seq-quiz-uml/)

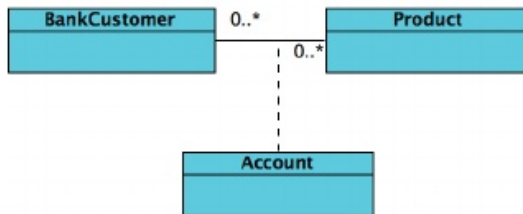
Quiz Q2 + Q5 – Q7

# Cheat sheet class diagram

Previously on: <http://www.code-meets-design.de/wp-content/uploads/2013/07/uml-classdiagram-cheat-sheet.pdf>



The Account belongs to one Bank. The Bank contains 0 to infinite Accounts



This type is often use in n to m relationships. The „Account“ chains the „Product“ with the



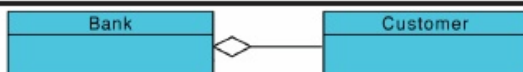
The class „KitchenAppliance“ implements the interface



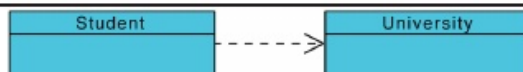
The account is part of the bank and can't exists without



The „Child“ extends the „Parent“ and contains every



The „Customer“ is part of the „Bank“, but the customer



The „Student“ dependents on the „University“