Chapter 3: UML

[Note, in WS 20/21 we skip 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

Real World: University

Conceptual Modeling

Students

Lectures

StudNr
Name

attend

LectureNr
Title

Students

Lectures
Modeling a small example application: UML

**Students**
- +StudNr : int
- +Name : String
- +Semester : int
- +GPA() : float
- +SumWeeklyHours() : short

**Lectures**
- +LectureNr : int
- +Title : String
- +WeeklyHours : int
- +NumberAttendees() : int
- +FailureRate() : float

**Attendee**
- 1..*

**Successor**
- *

**requires**
- *

**attend**
- *
Data modelling with UML

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

- Also other useful diagrams: state chart, activity, sequence..
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 analogously 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:
• “belongs-to” or “has”
• Multiple owners

Composition:
• “part-of”
• Special case of Aggregation
• Existence dependent
• Exactly one owner
### Multiplicity: Example 1

- A **Mentor** can have an arbitrary amount of **Student**
- A **Student** might have 0 or 1 **Mentor**

- Association type: Regular association (or aggregation)
Multiplicity: Example 2

- A **Car** has 4 **Wheels**
- A **Wheel** belongs to one **Car**
- Association type: Aggregation (or composition)
Multiplicity: Example 3

- A **Building** has at least 1 **Room**
- A **Room** belongs to exactly 1 **Building**
- Association type: Composition
Multiplicity: Example 4

- A Person has any number of friends
- Association type: Regular association
Multiplicity: Example 5

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

Class A

+ Att1
+ Att2
+ op()

Class B

+ Att1
+ Att2
+ op()

Class C

+ Att1
+ Att2

... 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

Students

+StudNr: int
+Name: string
+Semester: int

+GPA(): float
+SumWeeklyHours(): short

Exams

+Examinee

1

pass

* 

+Grade: decimal
+Date: date

+move()

* 

+Subject

1

... 

+Examiner

1 

...
Quiz UML

From the Stanford MOOC:


Quiz Q2 + Q5 – Q7
Cheat sheet class diagram


- **Bank** to **Account**: The Account belongs to one Bank. The Bank contains 0 to infinite Accounts.

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

- **KitchenAppliance** to **Socket**: The class „KitchenAppliance“ implements the interface

- **Bank** to **Account**: The account is part of the bank and can't exits without

- **Parent** to **Child**: The „Child“ extends the „Parent“ and contains every

- **Bank** to **Customer**: The „Customer“ is part of the „Bank“, but the customer

- **Student** to **University**: The „Student“ dependents on the „University“