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
  - StudNr
  - Name
- Lectures
  - LectureNr
  - Title

Attend
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

**LectureNr**
- *

**Title**
- *

**WeeklyHours**
- *

**NumberAttendees()**
- *

**FailureRate()**
- *
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 ..
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
• 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 \( \text{(min, max)} \)-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
• A **Mentor** can have an arbitrary amount of **Student**
• A **Student** might have 0 or 1 **Mentor**

• Association type: Regular association (or aggregation)
• 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
A **Person** has any number of friends

**Association type:** Regular association
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

\[\cdots\]\n
1 +Examiner

\[\cdots\]\n
Examinee

move()
### Studenten
- `MatrNr : int`
- `Name : String`
- `Semester : int`
- `Notenschnitt() : float`
- `SummeWochenstunden() : short`

### Vorlesungen
- `VorlNr : int`
- `Titel : String`
- `SWS : int`
- `AnzHörer() : int`
- `DurchfallQuote() : float`

### Prüfungen
- `Note : Decimal`
- `Datum : Date`
- `verschieben()`

### Assistenten
- `Fachgebiet : String`
- `Gehalt() : short`

### Professoren
- `Rang : String`
- `Notenschnitt() : float`
- `Gehalt() : short`
- `Lehrstundenzahl() : short`

### Angestellte
- `PersNr : int`
- `Name : String`
- `Gehalt() : short`
Quiz UML

From the Stanford MOOC:


Quiz Q2 + Q5 – Q7
Cheat sheet class diagram


1. 

Bank

Account

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

2. 

BankCustomer

Product

This type is often used in n to m relationships. The "Account" chains the "Product" with the

3. 

KitchenAppliance

<<Interface>>

Socket

The class "KitchenAppliance" implements the interface

4. 

Bank

Account

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

5. 

Parent

Child

The "Child" extends the "Parent" and contains every

6. 

Bank

Customer

The "Customer" is part of the "Bank", but the customer

7. 

Student

University

The "Student" depends on the "University"