# Informatik 1

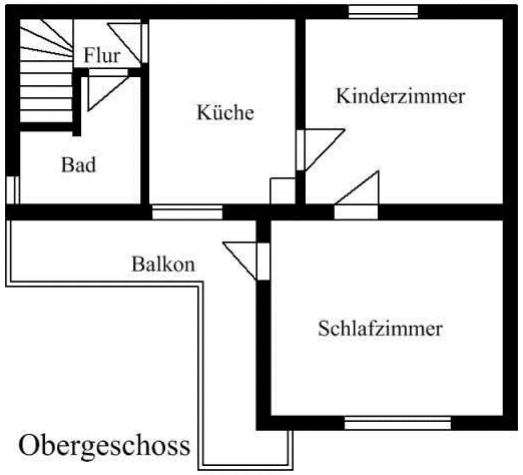
Unified Modeling Language

# Inhalt

- Aktivitätsdiagramm
- Klassendiagramm
- Objektdiagramm
- Paketdiagramm

# Unified Modeling Language (UML)

- Grafische Notation zur konzeptionellen Beschreibung (Entwurf, Design) und Dokumentation von Software
- **Analog Bauzeichnungen**
- Ursprung durch Grady Booch, Ivar Jacobson und James Rumbaugh bei Rational Software (1997)
- De facto Standard in der Softwareentwicklung
- Juni 2015 UML 2.5
- Reichlich Entwicklerwerkzeuge vorhanden
- Verschiedene Diagrammtypen
  - Verhaltensdiagramm: z.B. **Aktivitätsdiagramm**
  - Strukturdiagramm: z.B. **Klassen-, Objekt-, Paketdiagramm**

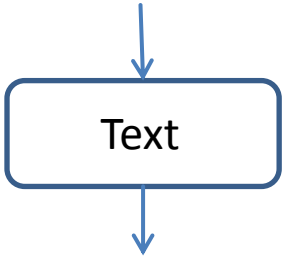


# Aktivitätsdiagramm

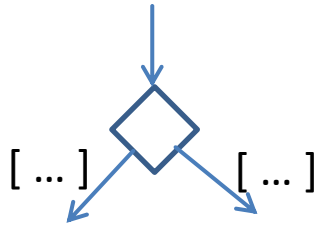
Startknoten

Kontrollkante

Endknoten

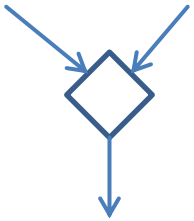



Aktion




Entscheidungsknoten

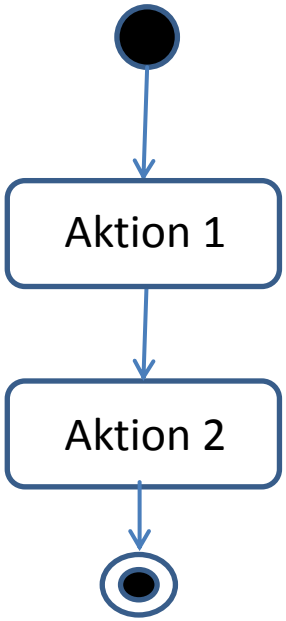
[ ] enthält eine logische Bedingung
Eine der Bedingungen muss zutreffen können



Verschmelzungs-
knoten

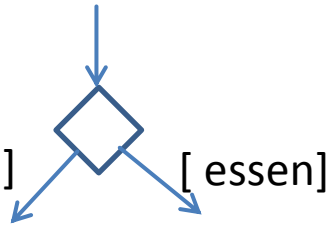
# Aktivitätsdiagramm

- Ein Startknoten
  - Ablauf der Aktivität fängt hier an
- Ein Endknoten
  - Aktivität endet hier
- Aktion
  - Etwas wird getan: durch geeigneten Text beschreiben
- **Zeitlicher** Ablauf zwischen Aktionen durch Kontrollkanten angeben
- Beispiel:
  - Zur Mensa essen gehen

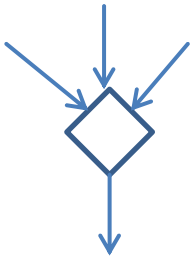


# Aktivitätsdiagramm

- Entscheidungen mit Verzweigungsknoten
  - Essen oder nicht Essen
  - Bedingungen umgangssprachlich in [ ] an den Kanten schreiben
  - Mindestens eine Bedingung muss wahr werden können
  - Mehr als zwei ausgehende Kanten möglich




- Kontrollflüsse zusammenfassen mit Verschmelzungsknoten
  - Mehr als zwei eingehende Kontrollflüsse möglich

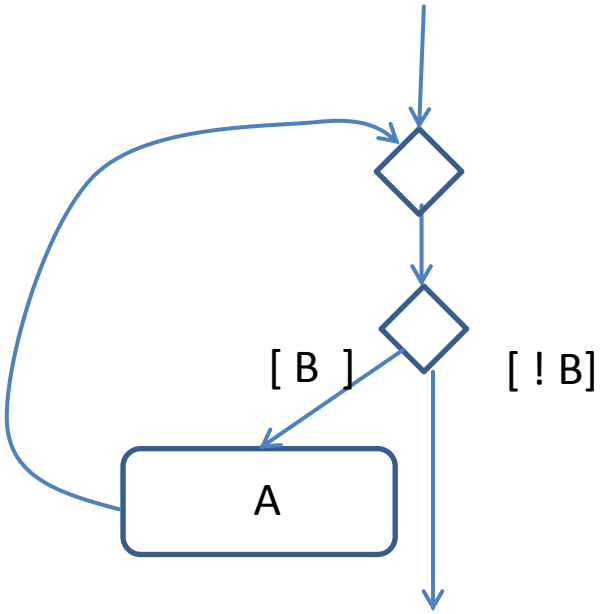


- Beispiel:
  - Zur Mensa essen gehen und entscheiden, ob man ein Essen dort essen will oder nicht

# Aktivitätsdiagramm

- Wiederholungen
  - Mit Entscheidungs- und Verschmelzungsknoten
  - z..B. wie while-Schleife

    while (B) { A }

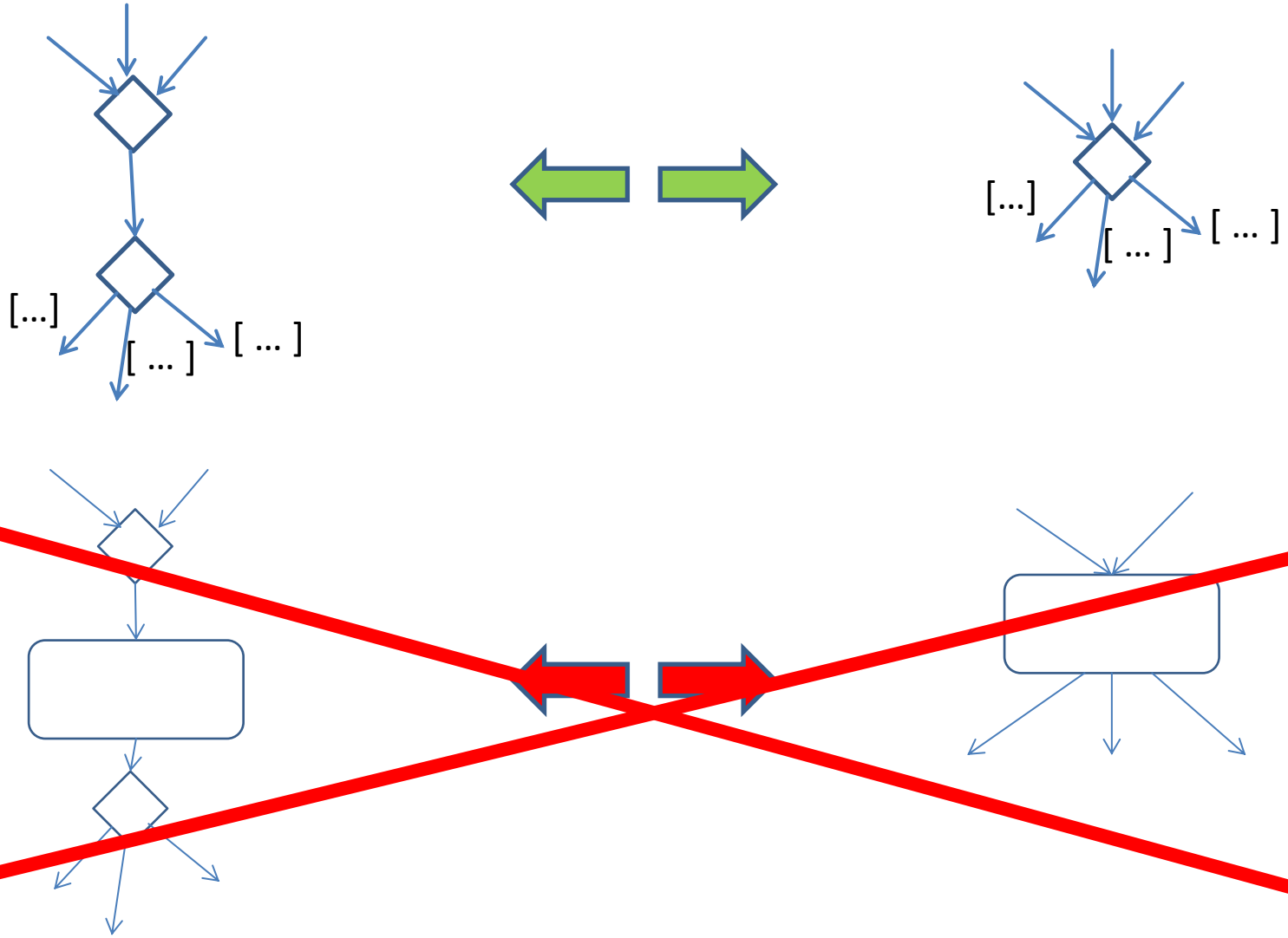



- Beispiel
  - Wieder ein Essen, wenn man weiterhin hungrig ist

# Aufgabe

- Aktivitätsdiagramm für die beiden folgenden Sachverhalte erstellen
  1. Der Krug geht so lange zum Brunnen, bis er bricht.
  2. Vor dem Überqueren der Straße sich durch links und nach rechts schauen vergewissern, dass die Straße frei ist. Falls die Straße nicht frei, ist etwas warten und dann wieder versuchen, die Straße zu überqueren.

# Abkürzungen



# Aktivitätsdiagramme

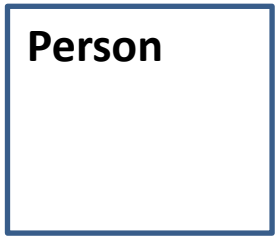
- Verwenden
  - Um umgangssprachliche Text und Aufgabenbeschreibungen genauer zu formulieren
  - Umgangssprache ist mehrdeutig
  - Aktivitätsdiagramme werden **eindeutiger**
- Nicht verwenden
  - Wenn Aktionen einzelne Programmanweisungen darstellen
  - Abstrakte Aktionen können dann zum Beispiel, mit einer Methode implementiert und der Schrittweisen-Verfeinerung weiter detailliert werden

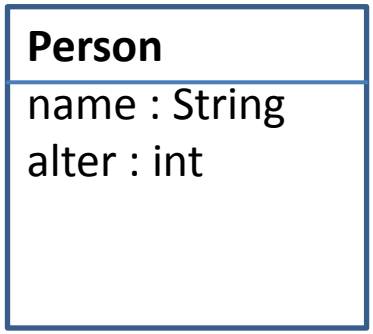
# Klassendiagramm

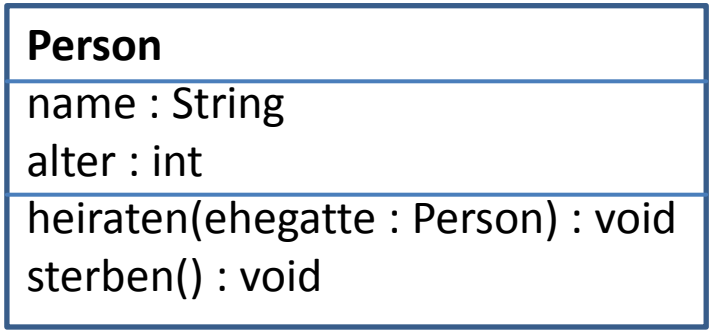
- Klassen: Mengen von Objekten
- Objekte: Abstraktionen realer „Dinge“ (reale Objekte)
  - der *Stuhl*, auf dem ich gerade sitze
  - die *Tafel*, auf der ich schreibe
  - die *Fenster*, in diesem Raum
  - das *Auto*, welches ich gerade fahre
  - die Zahl *Acht*
- Die Menge aller Stühle wird durch eine Klasse beschrieben
- Objekte
  - werden nicht vollständig mit Klassen beschrieben
  - nur Eigenschaften und Verhalten, welches für den Anwendungszweck benötigt wird (z.B. Programm)
- Verhalten( Objektmethoden, Nachrichten): Objekte können etwas tun, z.B. Auto *fährt*, Fenster *geht auf*
- Eigenschaften (Objektattribute) : Farbe, Gewicht, Größe, Füllstand

# Klassendiagramm

- Die Klasse wird als Rechteck gezeichnet
- Name der Klasse
  - Fett
  - Oben links im Rechteck
  - beschreibt, welche Objekte, die Menge enthält (Einzahl verwenden)
- Eigenschaften
  - Horizontale Linie unterhalb Klassename
  - werden mit Namen und Datentyp aufgezählt.
  - Semikolon zwischen Namen / Typ
- Verhalten
  - Horizontale Linie unterhalb Eigenschaften
  - Mit Namen, Parameter und Rückgabetyp
- Alle Angaben bis auf Klassennamen sind optional
  - Eigenschaft ohne Datentyp möglich
  - Verhalten ohne Parameter oder Rückgabetyp

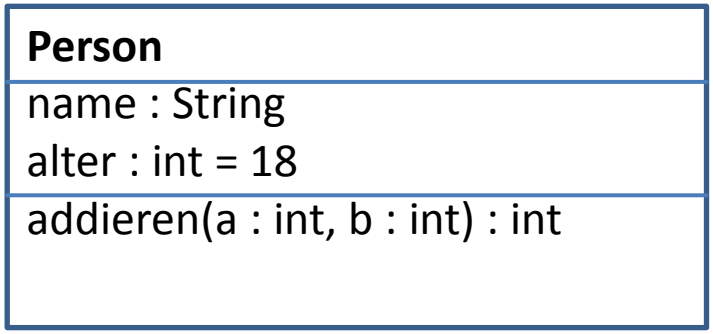


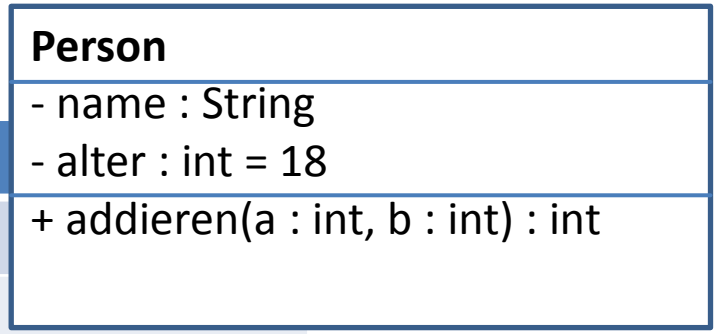
# Klassendiagramm

- Klassen beschreiben Wertemengen
  - Substantiv für Klasse verwenden
  - Klassen sind Datentypen
  - String ist Klasse, kann als Typ verwendet werden
- Eigenschaften / Objektattribute
  - Verhalten sich wie Variablen
  - Substantiv verwenden
  - Initiale Werte mit = hinter Datentyp möglich
- Verhalten / Objektmethoden
  - machen etwas
  - Verb verwenden
  - Mehrere Parameter mit Komma trennen
- UML hat eigenes Typsystem
  - Informatik 1: Java Datentypen verwenden



# Klassendiagramm

- Sichtbarkeit
  - Analog Klassenvariablen- und Methoden in Java
  - Verhindert in UML die Verwendung von Attributen und Methoden in anderen Diagrammen
  - Objektattribute private (Vererbung auch protected)
  - Methoden public oder private




| Java | UML | |
|---|---|---|
| public | + | überall sichtbar |
| private | - | nur in Klasse sichtbar |
| protected | # | Innherhalb Vererbungshierarchie (bei Java zusätzlich Paketsichtbarkeit) |
| | ~ | Paketsichtbarkeit |

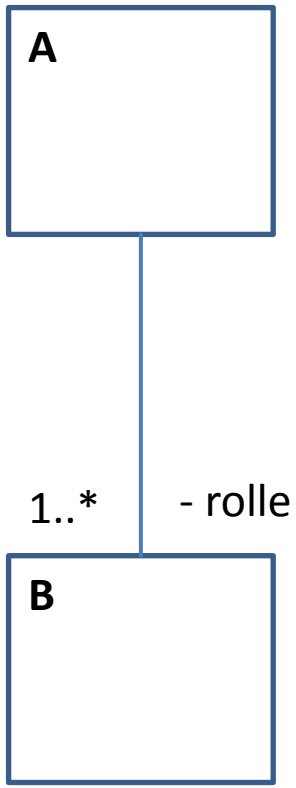
# Klassendiagramm

- Klassenvariablen und -methoden
  - müssen unterstrichen werden

| **Erde** |
| --- |
| - UMFANG_AEQUATOR : double = 40075.017<br>- bahngeschwindigkeit : double = 29.78 |
| + bahngeschwindigkeitErhoehen(prozent : double) : void |

# Klassendiagramm

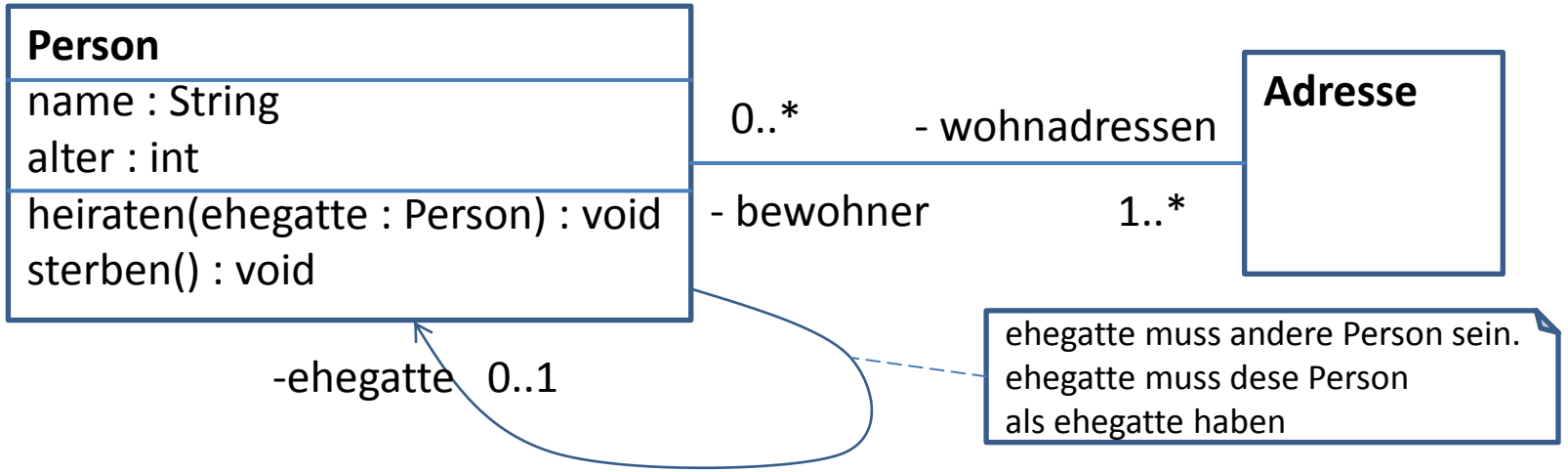
- Ein Person kann mehrere Wohnadressen besitzen
  - Mit endlich vielen Objektattributen nicht repräsentierbar
- Beziehungen (Assoziationen) verwenden
  - Linie zwischen zwei Klassen A und B
  - Anzahl Werte (**Häufigkeit**, Multiplizität, Kardinalität) von B, die zu A gehören, am Ende der Linie bei B schreiben
  - **Rollennamen** analog einer Objektvariablen inklusive Sichtbarkeit hinzufügen
- Häufigkeiten können auch in [ ] hinter dem Datentyp von Objektattributen verwendet werden



# Klassendiagramm

| Häufigkeit | Beschreibung |
|---|---|
| 1 | Genau eins |
| 6 | Genau sechs |
| 1, 2, 5 | Genau eins, zwei oder fünf |
| 5..8 | Fünf bis acht |
| 0..1 | Keine oder genau eins (d.h. Wert ist optional) |
| * | endlich viele, auch 0 |
| 2..* | mindestens zwei bis beliebig viele |

# Klassendiagramm




- Lesen als: „Eine Person **hat** eine oder beliebig viele Adressen, welche die Rolle von Wohnadressen einnehmen.“
- Oder kürzer: „Eine Person hat mindestens eine Wohnadresse“
- Rückrichtung: „Eine Adresse hat beliebig viele Bewohner.“
- **Richtung** mit Pfeil möglich:
  - Eine Person hat höchstens eine Person als Ehegatten
  - Umgekehrte Richtung ist dann nicht spezifiziert
- Zusätzliche Informationen mit **Notiz**

# Aufgabe

- Objekt-orientierte Analyse und Design (OOAD)
  - Methodik, um Anforderungen zu analysieren und einen ersten Entwurf zu erstellen
- Methodik für Klassendiagramme
  - Gegeben: Text mit Beschreibung eines Sachverhalts
  - **Substantive** geben Hinweise auf **potentielle Klassen oder Objektattribute**
  - **Verben** geben Hinweise auf **potentielle Objektmethoden**
  - Verschiedene Verben oder Substantiv können gleiches bezeichnen
- Designentscheidungen treffen!
  - Sachverhalte eindeutig festzulegen
  - geeignete Bezeichner und Datentypen wählen
  - Beziehungen, Rollennamen und Häufigkeiten festlegen
  - Entscheidungen können sich im späteren Verlauf als falsch erweisen
  - Offensichtliche Designentscheidungen zuerst treffen
- Entscheidungen schränken die Menge der möglichen Objekte ein

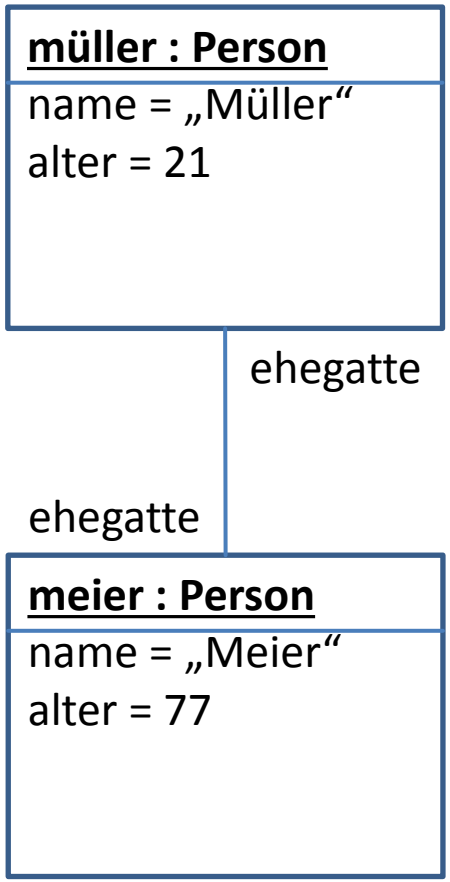
# Aufgabe

**Geben Sie ein Klassendiagramm für folgenden Sachverhalt an**

Ein Stausee hat eine Wassermenge in Kubikmetern und einen Wasserstand in Metern. Der Stausee wird von einem Staudamm gestaut. Der Staudamm besitzt mehrere Turbinen. Jede Turbine hat eine Maximalleistung in Watt pro Stunde. Eine Turbine kann an oder aus sein. Ein Turbine kann auch ausfallen. Der Stausee kann überlaufen, wenn der Wasserstand zu hoch wird.

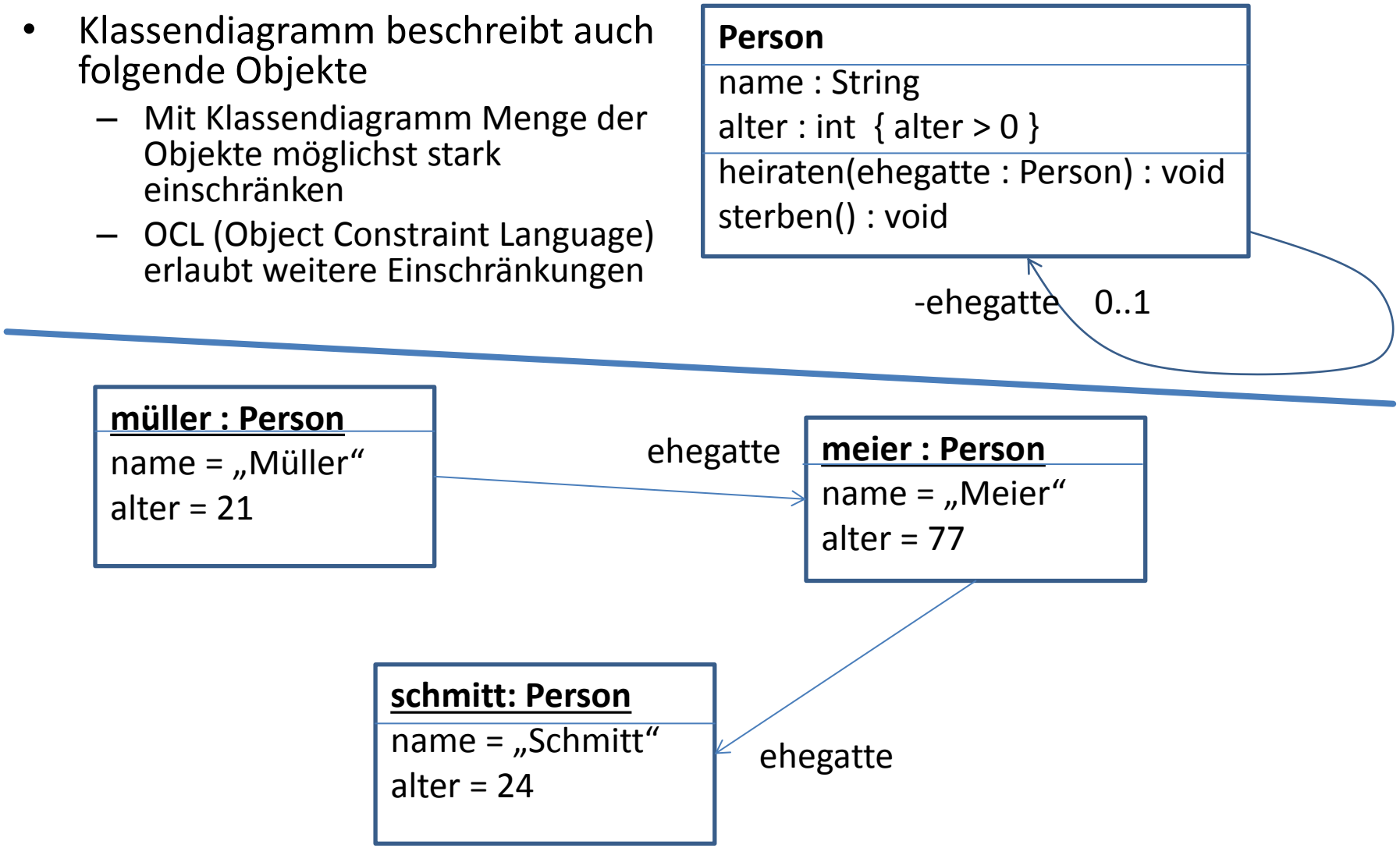
# Objektdiagramm

- Eigener Diagrammtyp
  - wird selten verwendet
  - eingeschränkte Darstellung der Werte einzelner Objekte
  - kann Schnappschuss einer speziellen Situation beschreiben
  - Darstellung von Objekten wird hauptsächlich in Sequenzen- und Kollaborationsdiagrammen verwendet
- Darstellung
  - Ein Rechteck pro Objekt
  - Name des Objekte (optional) gefolgt vom Datentyp (Klasse), Semikolon dazwischen, alles unterstreichen
  - Ausgewählte Eigenschaften mit Namen, Gleichheit und Wert angeben
  - Beziehung zu anderen Objekten mit Strich und Rolle (optional), Richtung mit Pfeil möglich



# Objektdiagramm

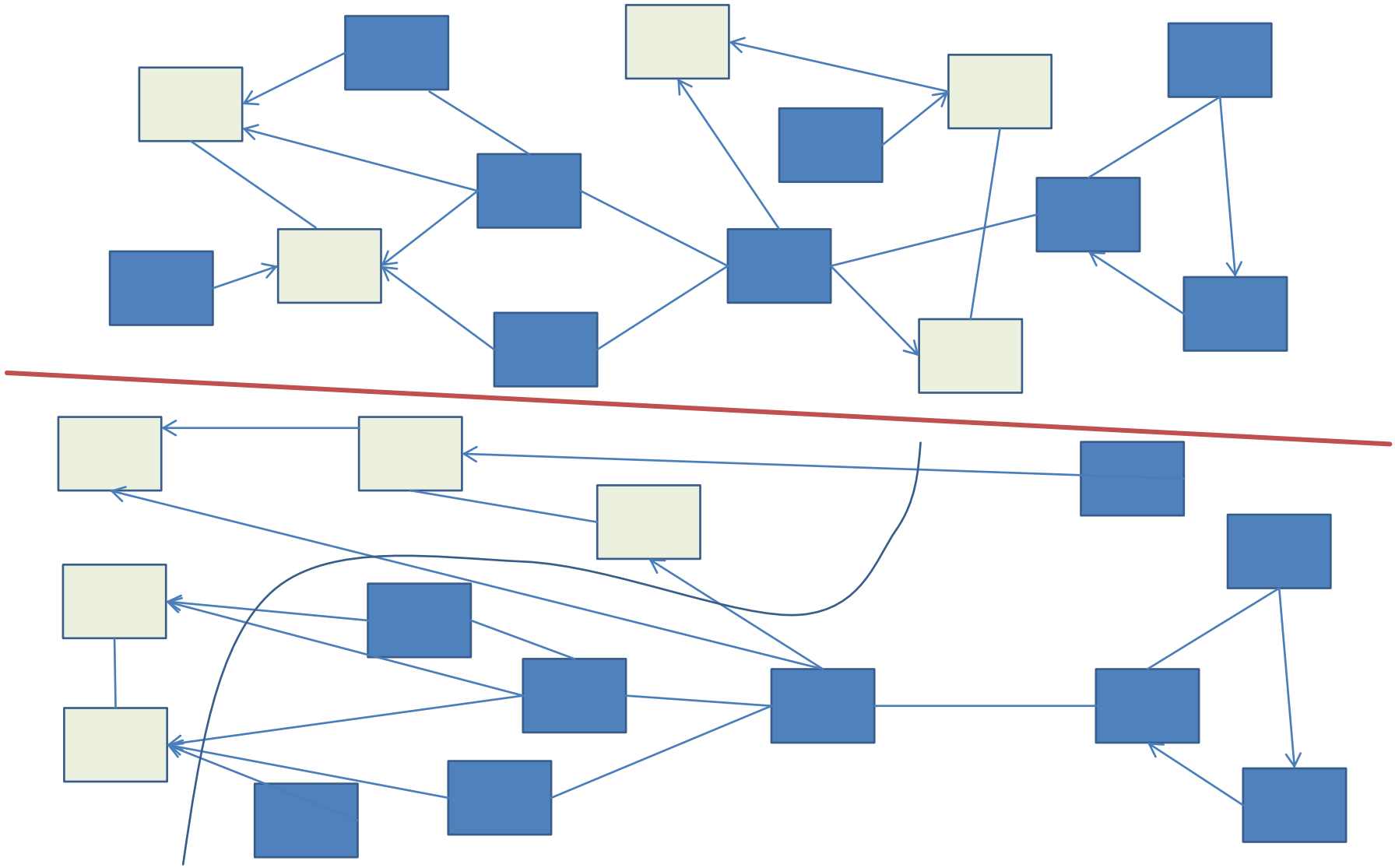
- Klassendiagramm beschreibt auch folgende Objekte
  - Mit Klassendiagramm Menge der Objekte möglichst stark einschränken
  - OCL (Object Constraint Language) erlaubt weitere Einschränkungen



# Paketdiagramm

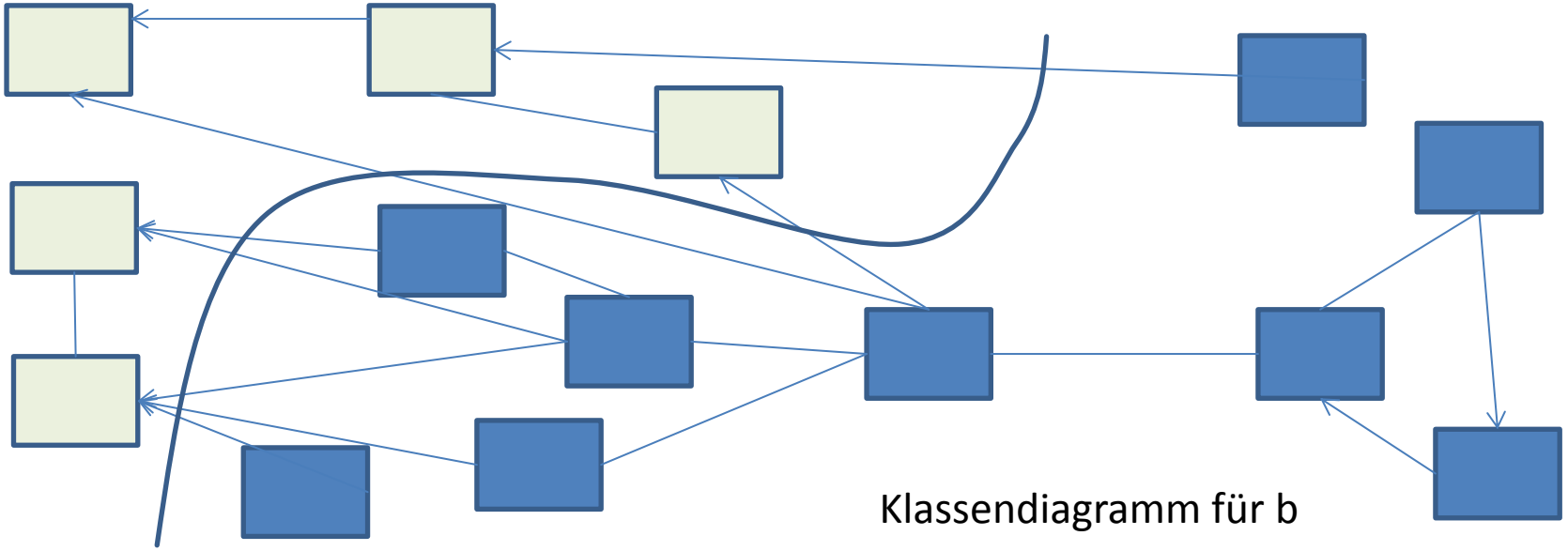
- Ein Paket ist eine Menge von Klassen und Paketen
  - Pakete organisieren Klassen in einer Hierarchie analog Dateihierarchie
  - Abhängigkeiten zwischen Paketen können definiert werden: z.B. ein Paket darf keine Klassen aus einem anderen Paket verwenden
  - UML: Dateiordnersymbol mit Namen
  - Einem Paket kann ein Paket- oder Klassendiagramm zugeordnet werden
- Pakete unterteilen Software in kleinere Teile oberhalb der Klassenebene
  - Reduzierung von Komplexität und Abhängigkeiten

# Paketdiagramm



# Paketdiagramm


Klassendiagramm für a



Klassendiagramm für b

# Paketdiagramm

- Paketdiagramm können früh im Entwurf verwendet werden, wenn Abhängigkeiten bekannt sind
- Beispiel: (private) Krankenversicherung
  - Versicherter kennt Arzt und Versicherung (und umgekehrt)
  - Versicherung kennt nur Rechnungen vom Arzt