Unified Modelling Language

Klassendiagramm

– Objekt, Klasse, Operation

Überblick

1. Objekt
2. Klassen
3. Attribute
4. Operationen

1. Objekt

1. Definition
2. Notation
3. Beispiel
4. Objektdiagramm

Definition

- modelliert
  - Dinge (z.B. Auto)
  - Personen (z.B. Student)
  - Begriffe (z.B. Krankheit)
  - …

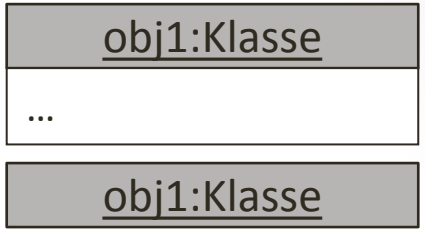
Definition

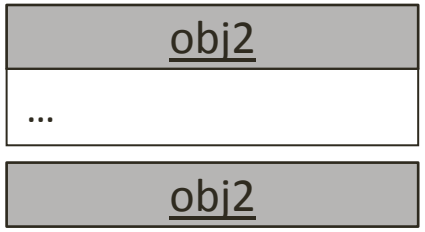
- eindeutig indentifizierbar
- Zustand (state)
  - Attribute und Attributwerte
  - Beziehungen zu anderen Objekten
- Verhalten (behaviour)
  - verfügbare Operationen

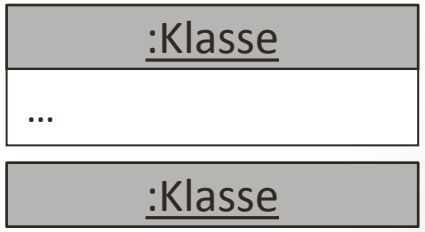
Notation

Notation von Objekten

- Rechteck, unterteilt in „compartments“
- 1. Compartment:
  - objektname:Klasse
  - objektname
  - :Klasse (anonymes Objekt)





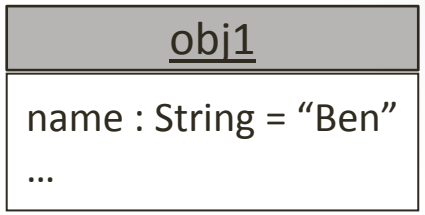


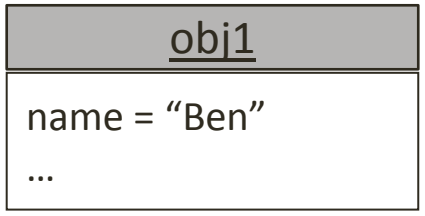
- Objektnamen kleingeschrieben !
- Klassennamen großgeschrieben !!
- beides unterstrichen !!!

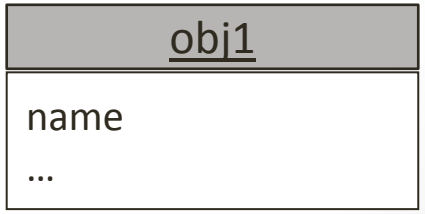
Notation

Notation von Objekten

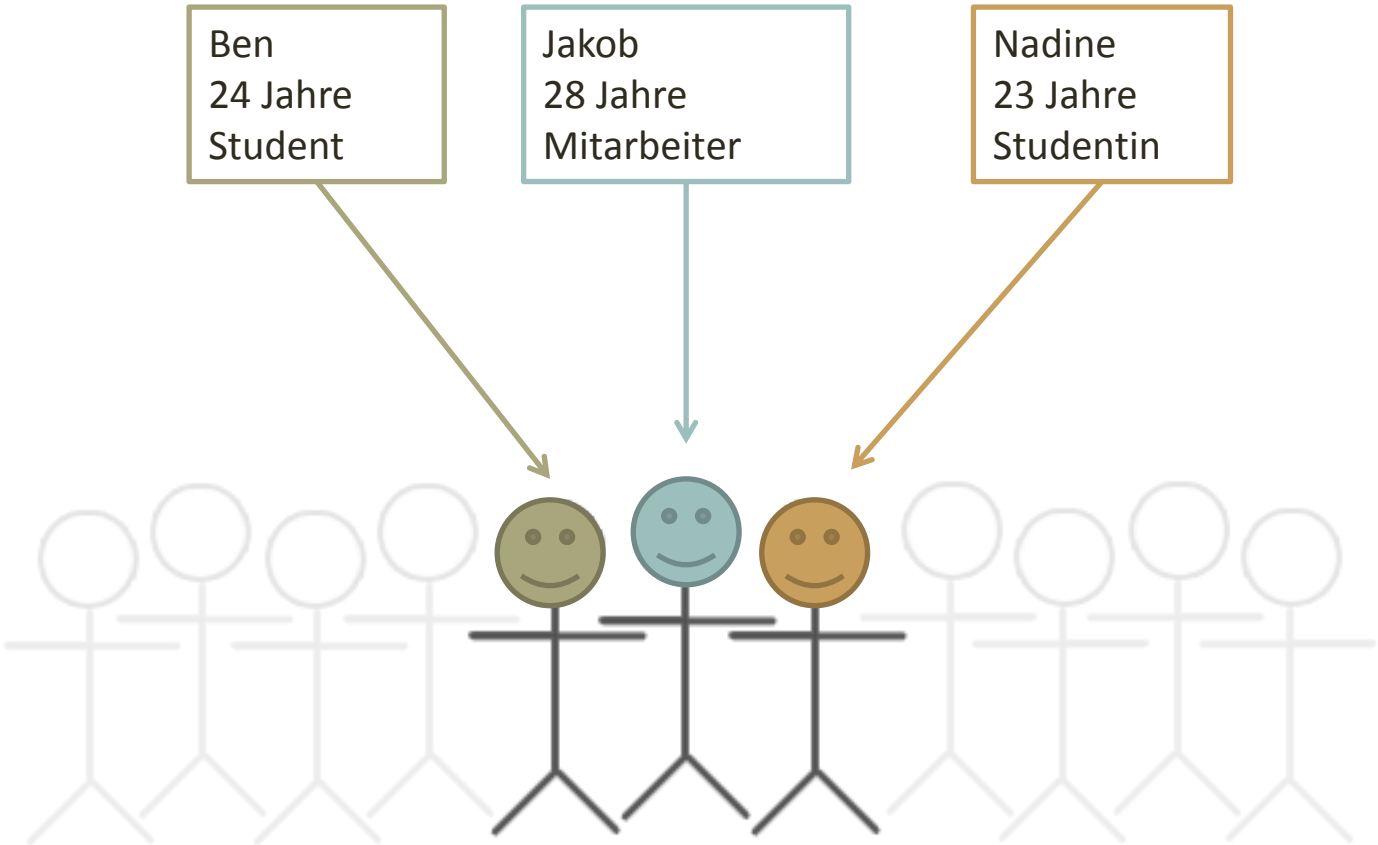
- 2. Compartment:
  - Optional
  - Im Kontext relevante Attribute und Werte
  - attribut : Typ = Wert
  - attribut = Wert
  - attribut



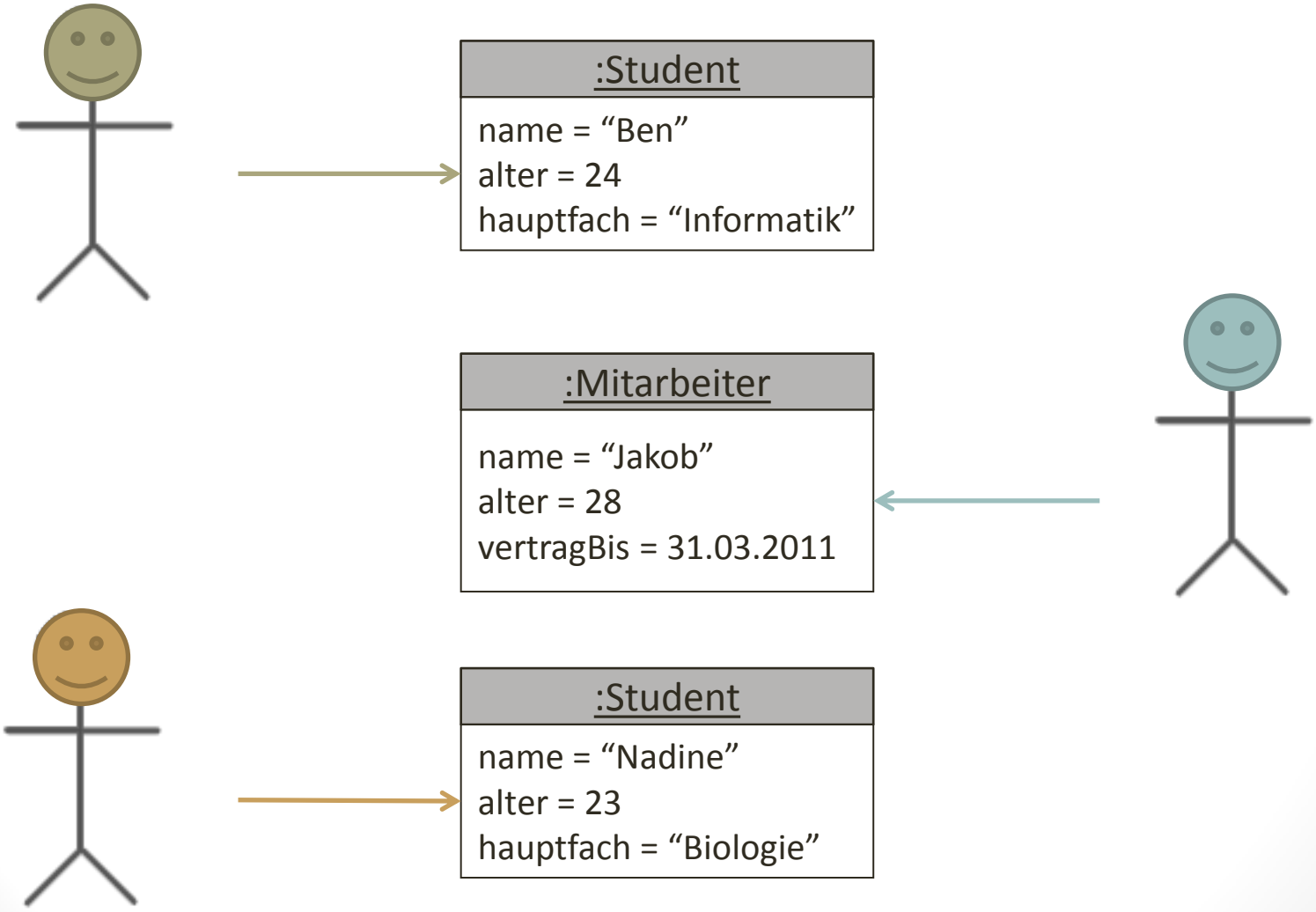




Beispiel

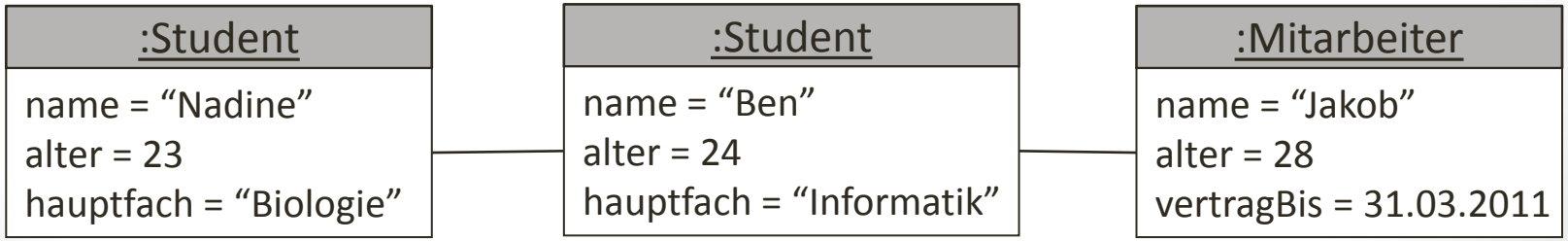

Ben
24 Jahre
Student
Jakob
28 Jahre
Mitarbeiter
Nadine
23 Jahre
Studentin

Beispiel


:Student
name = “Ben”
alter = 24
hauptfach = “Informatik”
:Mitarbeiter
name = “Jakob”
alter = 28
vertragBis = 31.03.2011
:Student
name = “Nadine”
alter = 23
hauptfach = “Biologie”

Objektdiagramm

- Momentaufnahme eines Systems:
  - Objekte (Attribute und Attributwerte)
  - Beziehungen zwischen Objekten



2. Klassen

2. Klassen

1. Definition
2. Klassendiagramm
3. Notation
4. Abstrakte Klassen

Definition

- Menge von Objekten mit gleicher Struktur und gleichem Verhalten
  - Attribute beschreiben Struktur
  - Operationen beschreiben Verhalten

- Klassennamen: möglichst aussagekräftige Substantive

Klassendiagramm

- stellt Klassen und deren Beziehungen dar

- Klassen als Rechteck dargestellt
- unterteilt in (beliebig viele) Compartments
- Inhalt der Compartments prinzipiell frei wählbar

Konvention:

Klassenname
Attribute
Operationen
…

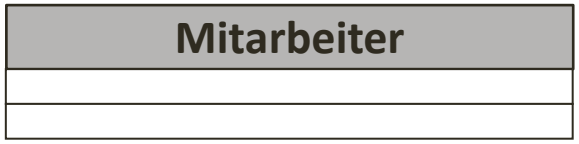
fehlende Compartments bedeuten nicht fehlende Informationen; verschiedene Sichten möglich !!!

KlasseA
attribut_1 attribut_2

KlasseA
attribut_x attribut_y

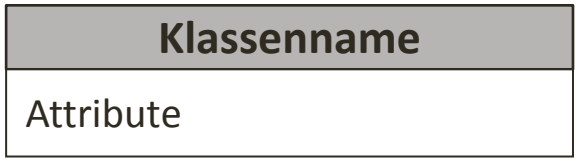
Notation

- 1. Compartment: Klassenname
  - Subsantiv im Singular
  - beginnt mit Großbuchstabe
  - zentriert, fett gedruckt
  - innerhalb des Pakets eindeutig
  - bei Bedarf qualifizierter Name Paket::Klassenname

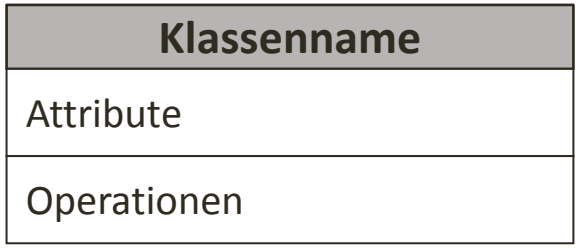


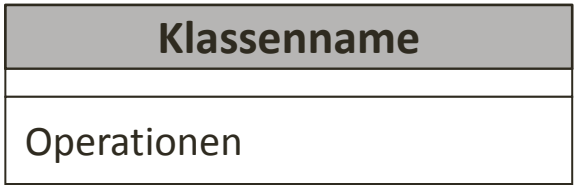
Notation

- 2. Compartment: Attribute



- 3. Compartment: Operationen





- 4+ Compartment: beliebige Informationen

Abstrakte Klassen

- können nicht instanziiert werden
- Eigenschaften und Verhalten verschiedener Arten von Objekten
- nicht in der Klasse selbst implementiert
- kursiver Klassenname oder Schlüsselwort abstract

Maschine

Maschine
{abstract}

3. Attribute

3. Attribute

1. Definition
2. Notation
3. Klassenattribute

Definition

- Eigenschaften von Klassen
- sind getypt

- gleiche Attribute in allen Instanzen einer Klasse
- Attributwerte können sich unterscheiden

Notation

- konventionell im zweiten Compartment
- Syntax:

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

- mit Ausnahme von Name sind alle Angaben optional

Notation . Sichtbarkeit

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Sichtbarkeit (visibility)

- + public
- ~ package
- # protected
- - private

Klasse
+ attr1 ~ attr2 # attr3 - attr4

- in OOP oftmals Attribute private
- Zugriff über Operationen

Notation . Abgeleitete Attribute

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Abgeleitete Attribute (derived attributes)

- können jederzeit aus anderen Attributwerten der Klasse errechnet werden
- keine neuen Informationen

Student
Geburtsdatum / alter

Würfel
länge breite höhe / volumen

Notation . Name

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Name:

- Substantive (Konvention)
- beginnt mit Kleinbuchstaben
- danach (fast) alle Zeichen erlaubt
- innerhalb einer Klasse eindeutig

Baum
umfang des Stamms art höhe

nicht erlaubt:

Klasse
xy : Integer xy : Integer

Klasse
xy : Integer xy : Boolean

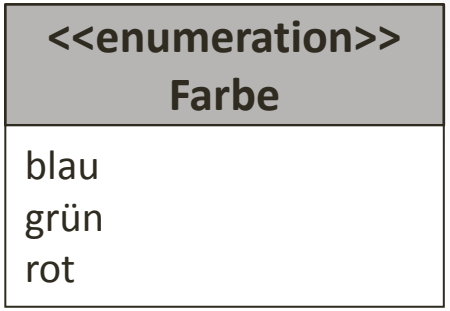
Klasse
x:y : Integer

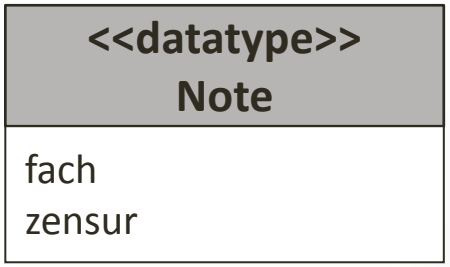
Notation . Typ

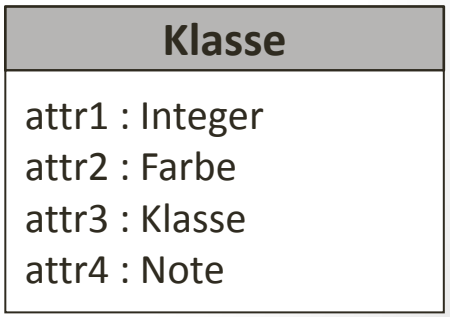
Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Typ:

- Datentypen
  - Werte besitzen keine Identität
  - Stereotype <<datatype>>
- Primitive Datentypen
  - besitzen keine Struktur (Attr./Oper.)
  - Stereotype <<primitive>>
  - Boolean / String / Integer / UnlimitedNatural
- Aufzählungstypen
  - Stereotype <<enumeration>>
- Klassen







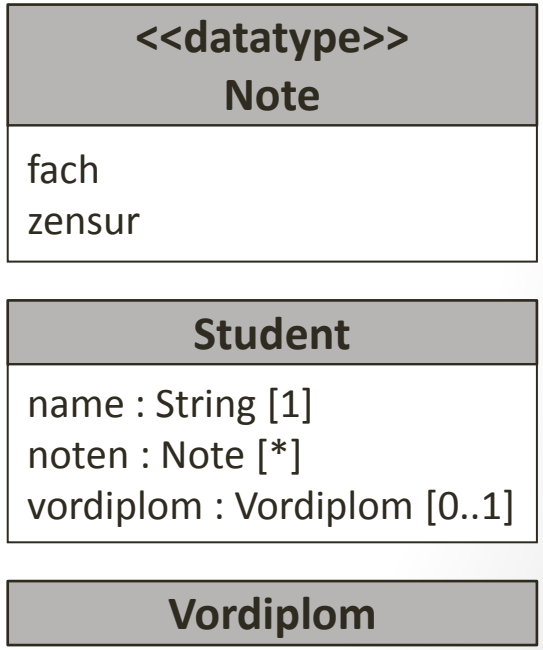
Notation . Multiplizität

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Multiplizität (multiplicity):

- mehrere Werte für ein Attribut

- [untere Schranke .. obere Schranke]
- Standard: [1..1] = [1]
- beliebig viele: [*] = [0..*]
- Optionalität: [0..1]
- allgemein: [m..n]



Notation . Anfangswert

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Anfangswert:

- Wert, der beim Erzeugen eines Objekts zugewiesen wird

Student
name : String fachsemester : Integer = 1

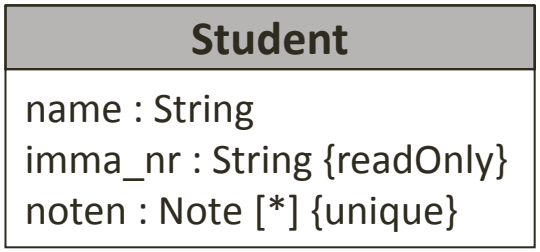
Notation . Eigenschaftswerte

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Eigenschaftswerte / Einschränkungen

Eigenschaftswerte:

- Spezielle Eigenschaften / Merkmale
  - {readOnly} : Attribut darf nicht verändert werden
  - {ordered} : Elemente sind geordnet *[1,2,3,3,6,6]*
  - {unique} : jedes Element muss einzigartig sein *[3,1,4,2,6]*
  - …



Notation . Einschränkungen

Sichtbarkeit / Name : Typ Multiplizität = Anfangswert {Einschränkungen}

Eigenschaftswerte / Einschränkungen

Einschränkungen (constraints):

- Zusicherungen, die immer erfüllt sein müssen
- Formulierung z.B. in OCL, aber auch natürliche Sprache

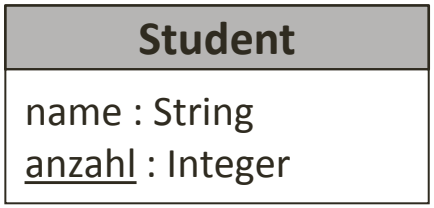
Student
name : String geburtsdatum : Datum immatrikulation : Datum { geburtsdatum < immatrikulation }

Klassenattribute

Klassenattribute / statische Attribute

- Attribut der Klasse
- existiert bevor Objekte instanziiert werden
- ein Attributwert für alle Objekte

- werden unterstrichen dargestellt



4. Operationen

4. Operationen

1. Definition
2. Notation
3. Eindeutigkeit
4. Abstrakte Operationen
5. Aktive Klassen

Definition

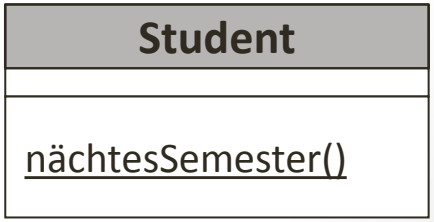
- beschreiben das Verhalten einer Klasse bzw. deren Objekte
- Operation = bestimmtes Verhalten, Signatur der Methode
- Methode = Implementation dieses Verhaltens

- alle Objekte einer Klasse haben die gleichen Operationen
- Operation kann auf Attribute zugreifen und diese ändern

- Möglichkeit mit Objekten zu interagieren

Definition

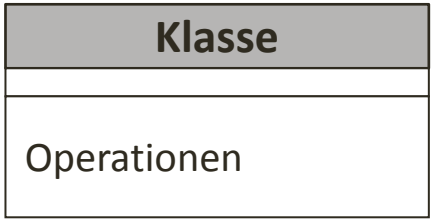
- Objektoperationen
  - werden auf existierendes Objekt angewendet
  - Manipulation der Attributwerte o.ä.
- Klassenoperationen
  - wird auf Klasse angewendet
  - z.B. Manipulation von Klassenattributen
  - z.B. Konstruktor (neues Objekt erzeugen, Initialisierungen)
  - unterstrichen



Notation

- üblicherweise im 2. Compartment
- Syntax:

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}



Notation . Sichtbarkeit / Name

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

Sichtbarkeit

- analog zu Attributen
- auch hier optional

Name

- analog zu Attributen
- sollte Verb enthalten (Konvention)

Notation . Parameter

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

Parameter

- Syntax:

Richtung Name : Typ [Multiplizität] = DefaultWert {Einschränkungen}

Notation . Parameter

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

Richtung Name : Typ [Multiplizität] = DefaultWert {Einschränkungen}

Richtung

- in (Standard)
- out
- inOut
- return

Notation . Parameter

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

Richtung Name : Typ [Multiplizität] = DefaultWert {Einschränkungen}

Name / Typ / Multiplizität / Einschränkungen

- analog zu Attributen

DefaultWert

- Wert des Parameters, falls kein Wert explizit übergeben wurde

Notation . Rückgabetyp

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

Rückgabetyp

- optional
- Datentyp / Klasse
- „void“, falls keine Rückgabe

Notation . Einschränkungen

Sichtbarkeit Name (Parameter) : Rückgabetyp {Einschränkungen}

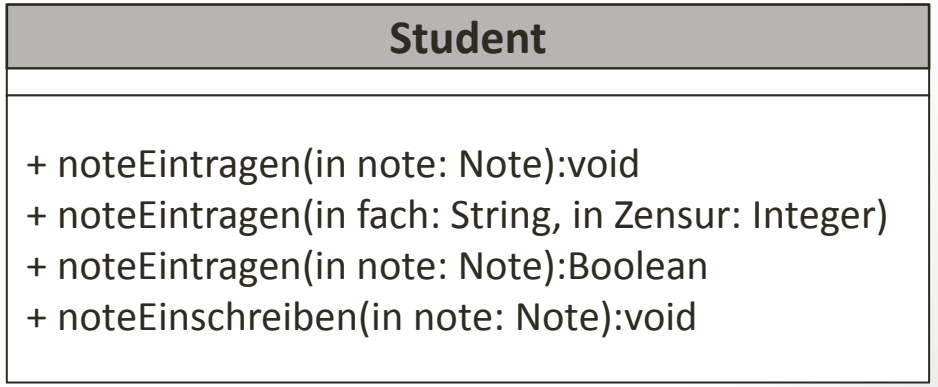
Einschränkugen

- Preconditions
  - zu erfüllende Bedingungen zur Aktivierung der Operation
- Postconditions
  - Zusicherung über Zustand des Systems nach der Ausführung
- Body conditions
  - Einschränkungen des Rückgabewerts

Eindeutigkeit

Eindeutigkeit

- Operationen müssen in einer Klasse eindeutig sein
- unterscheiden sich in
  - Name der Operation
  - Parameterliste (Typ)
  - Rückgabetyp



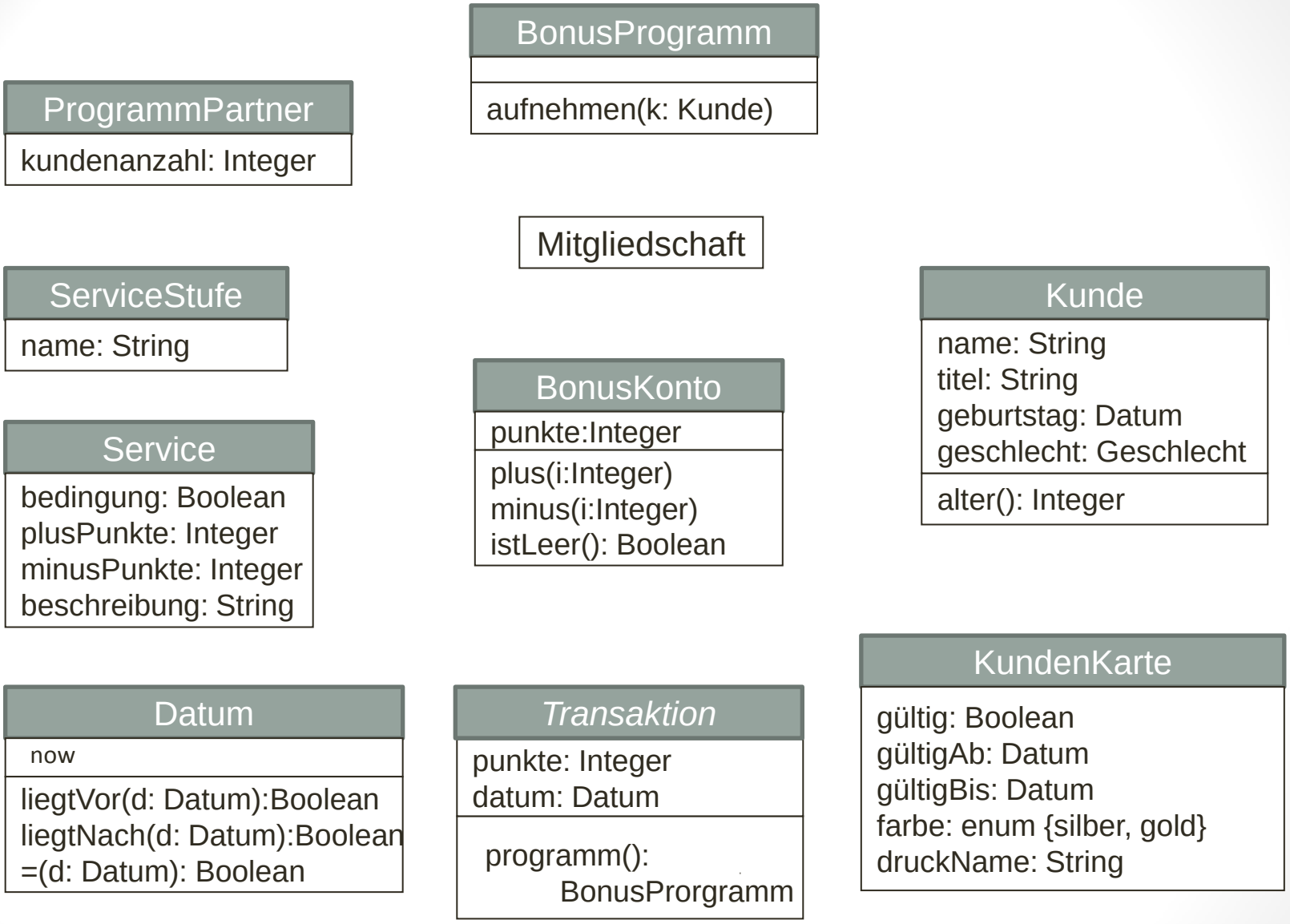
Abstrakte Operationen

Abstrakte Operationen

- in abstrakten Klassen
- es existiert keine Methode, nur die Signatur
- Operation kann nicht aktiviert werden

Aktive Klassen

- Objekte aktiver Klassen = aktive Objekte
- ohne externe Aktivierung aktiv
- als Prozess ausgeführt
- Aktivitätsdiagramm


BonusProgramm
aufnehmen(k: Kunde)
ProgrammPartner
kundenanzahl: Integer
Mitgliedschaft
ServiceStufe
name: String
Kunde
name: String
titel: String
geburtstag: Datum
geschlecht: Geschlecht
alter(): Integer
BonusKonto
punkte:Integer
plus(i:Integer)
minus(i:Integer)
istLeer(): Boolean
Service
bedingung: Boolean
plusPunkte: Integer
minusPunkte: Integer
beschreibung: String
KundenKarte
gültig: Boolean
gültigAb: Datum
gültigBis: Datum
farbe: enum {silber, gold}
druckName: String
Datum
now
liegtVor(d: Datum):Boolean
liegtNach(d: Datum):Boolean
=(d: Datum): Boolean
Transaktion
punkte: Integer
datum: Datum
programm():
BonusProrgramm

Quellen

Lehrbuch der Objektmodellierung

- Heide Balzert
- Spektrum, 2005

UML 2.0 in a Nutshell

- Dan Pilone, Neil Pitman
- O‘Reilly, 2005

UML 2 für Studenten

- Harald Störrle
- Pearson, 2005