TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de), Simon Ellmann (ellmann@in.tum.de)
http://db.in.tum.de/teaching/ss24/ei2/

Lösungen zu Blatt Nr. 2

Dieses Blatt wird am Montag, den 29.04.2024 besprochen.
Zugehörige Vorlesungsaufzeichnung: Vorlesung 3

Aufgabe 1: Typisierung

Typisieren Sie die nachfolgenden Pfadausdrücke wie in dem Beispiel in Abbildung 1. Geben Sie zusätzlich an, welchen Wert der gesamte Ausdruck hat.

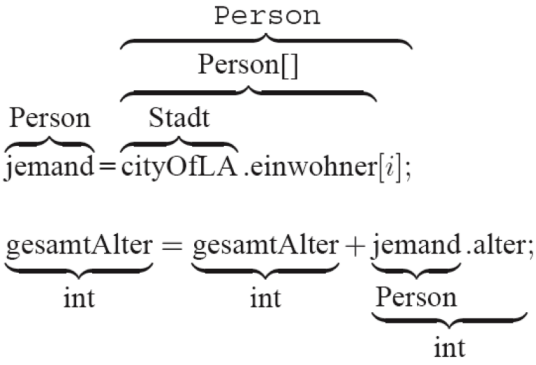

Abbildung 1: Typisierung von Pfadausdrücken

Achten Sie auf Konsistenz mit der Beispielimplementierung des Universitätsmodells. Die Objekte seien wie folgt instanziiert:

```
1 Professor sokrates = new Professor(2125, "Sokrates",
2                                    Professor.Rang.C4);
3 Vorlesung ethik = new Vorlesung(5041, "Ethik", 4, sokrates);
4 Assistent aristoteles = new Assistent(3003, "Aristoteles",
5                                       "Syllogistik", sokrates);
6 Student fichte = new Student(26120, "Fichte", 10);
7 Calendar termin = Calendar.getInstance();
8 Pruefung pruefung = new Pruefung(fichte, ethik, sokrates, termin);
9 int netto, jahre, nr, stunden;
```

a) `fichte.name`

b) `netto = sokrates.gehalt() - aristoteles.boss.steuern()`

c) `jahre = pruefung.pruefling.semester / 2`

d) `nr = ethik.dozent.persNr`

e) `aristoteles.boss.rang`

f) `stunden = pruefung.pruefungsstoff.dozent.lehrstunden`

Lösung:

a) `[[Student] String] => String`

b) `[int] = [[Professor] int] - [[[Assistent] Professor] int] => int`

c) `[int] = [[[Pruefung] Student] int] / [int] => int`

d) `[int] = [[[Vorlesung] Professor] int] => int`

e) `[[[Assistent] Professor] Rang] => Rang`

f) `[int] = [[[[Pruefung] Vorlesung] Professor] int] => int`

Aufgabe 2: Werte vs. Objekte

a) Nennen Sie mindestens drei Unterschiede zwischen Werten und Objekten.

b) Wann verwendet man Komposition und wann Aggregation?

c) Was ist der Unterschied zwischen Klassenattributen und Objektattributen?

d) Initialisieren Sie ein Array auf zwei verschiedene Arten mit den Zahlen 2, 3, 5, 7, 11, 13, 17, 19 und 42. Welche würden Sie bevorzugen?

Lösung:

a) Nennen Sie mindestens drei Unterschiede zwischen Werten und Objekten.

- Objekte haben eine Identität
- Objekte haben einen änderbaren Zustand (Attribute) und definieren Verhalten (Methoden) wohingegen Werte unveränderlich sind
- Objekte können mehrfach referenziert werden, Werte werden immer kopiert
- Objekte werden mit einem Konstruktor instanziiert und initialisiert
- Man kann eigene Objekttypen definieren, bei Wertetypen ist man auf die vordefinierten beschränkt (int, float, char, ...)
- ...

b) Wann verwendet man Komposition und wann Aggregation?

Die *Aggregation* kann für alle Teil/Ganzes-Beziehung verwendet werden, z.B. ein Gebäude besteht aus vielen Räumen, ein Aufsichtsrat besteht aus vielen Managern. Die *Komposition* ist dagegen eine spezielle Form der Aggregation. Sie beschränkt sich auf exklusive und existenzabhängige Teil/Ganzes-Beziehungen. Da ein Raum nur zu genau einem Gebäude gehören kann (Exklusivität) und nach dem Abriss dieses Gebäudes auch nicht mehr existiert (Existenzabhängigkeit), lässt sich diese Beziehung auch als Komposition darstellen. Nicht so beim Aufsichtsrat: Ein Manager kann in mehreren Aufsichtsräten sein (nicht exklusiv) und falls der Aufsichtsrat aufgelöst wird, existiert der Manager auch weiter (nicht existenzabhängig).

c) Was ist der Unterschied zwischen Klassenattributen und Objektattributen?

Ein Klassenattribut gehört wie der Namen schon sagt zu einer Klasse und hat für alle Objekte dieser Klasse immer den gleichen Wert. Von den Objektattributen hat jedes Objekt einer Klasse hingegen eine eigene Kopie. In diesen Attributen können sich verschiedene Objekte daher unterscheiden.

d) Initialisieren Sie ein Array auf zwei verschiedene Arten mit den Zahlen 2, 3, 5, 7, 11, 13, 17, 19 und 42. Welche würden Sie bevorzugen?

Die erste Variante verwendet den Standard-Konstruktor und viele einzelne Zuweisungen:

```
int[] array = new int[9];
array[0] = 2;
array[1] = 3;
array[2] = 5;
array[3] = 7;
array[4] = 11;
array[5] = 13;
array[6] = 17;
array[7] = 19;
array[8] = 42;
```

Die zweite Variante verwendet stattdessen den kürzeren „array initializer“:

```
int[] array = {2, 3, 5, 7, 11, 13, 17, 19, 42};
```

Aufgabe 3: Assoziationen in UML

Beschriften Sie die Assoziationen und Aggregationen im Klassendiagramm in Abbildung 3 (letzte Seite) mit den korrekten Funktionalitäten. Dabei sollte ein Student bis zu fünf Bücher ausleihen können, ein Mitarbeiter unbegrenzt viele.
Lösung: Siehe Abbildung 3 (letzte seite).

Aufgabe 4: Java

Laden Sie die Implementierung des Universitätsmodells von der Vorlesungswebseite (oder Moodle) herunter. Kompilieren Sie die Dateien und führen Sie *Test.java* aus.

a) Wir haben in Aufgabe 1 das Nettogehalt eines Professors bestimmt. Wir wollen dies nun als Methode implementieren. Zu welcher Klasse sollte diese am besten hinzugefügt werden? Implementieren Sie die Methode in dieser Klasse. Testen Sie die Methode anschließend in *Test.java*.

b) Erweitern Sie die Klasse *Student.java* um eine Methode, die alle Vorlesungen zurückgibt, bei denen der Student durchgefallen ist. Fügen Sie zu Testzwecken in *Test.java* einen Aufruf der Methode hinzu (und noch ein paar weitere Prüfungen). Überzeugen Sie sich, dass ihre Implementierung funktioniert.

Hinweis: Sie finden auf Panopto ein Video, in dem der Import des Universitätsschemas für IntelliJ gezeigt wird: Anleitung.
Lösung:

a) Wir haben in Aufgabe 1 das Nettogehalt eines Professors bestimmt. Wir wollen dies nun als Methode implementieren. Zu welcher Klasse sollte diese am besten hinzugefügt werden? Implementieren Sie die Methode in dieser Klasse. Testen Sie die Methode anschließend in *Test.java*.

Die Methode sollte in der Klasse *Angestellter* implementiert werden, da alle Angestellten über ein Gehalt verfügen und Steuern bezahlen müssen:

```
1  public class Angestellter {
2
3    ...
4
5    public int nettogehalt() {
6      return gehalt() - steuern();
7    }
8  }
```

b) Erweitern Sie die Klasse *Student.java* um eine Methode, die alle Vorlesungen zurückgibt, bei denen der Student durchgefallen ist. Fügen Sie zu Testzwecken in *Test.java* einen Aufruf der Methode hinzu (und noch ein paar weitere Prüfungen). Überzeugen Sie sich, dass ihre Implementierung funktioniert.

```
1  import java.util.Set;
2  import java.util.HashSet;
3  import java.util.Iterator;
4
5  public class Student {
6
7    ...
8
9    public Set<Vorlesung> durchgefallen() {
10     HashSet<Vorlesung> result = new HashSet<Vorlesung>();
11     for (Pruefung p : pruefungen) {
12       if (p.note > 4.0) {
13         result.add(p.pruefungsstoff);
14       }
15     }
16     return result;
17   }
18 }
```

Aufgabe 5: Bonus: UML Model

[[Dies ist ein Beispiel wie eine Aufgabe in der man ein UML Diagram zeichnen soll in einer Klausur aussehen könnte]]

Erstellen Sie anhand der folgenden Anforderungen ein UML Klassendiagramm mit Multiplizitäten für einen online Streaming Service:

- Es gibt Filme. Diese haben einen Titel und ein Erscheinungsdatum.
- Ein Film hat beliebig viele Sequels, aber nur maximal ein Prequel.
- Schauspieler haben einen Namen und ein Geschlecht.
- Schauspieler spielen in Filmen mit.
- Jeder Film hat zwischen 1 und 5 Genres.
- Genres haben einen Namen.

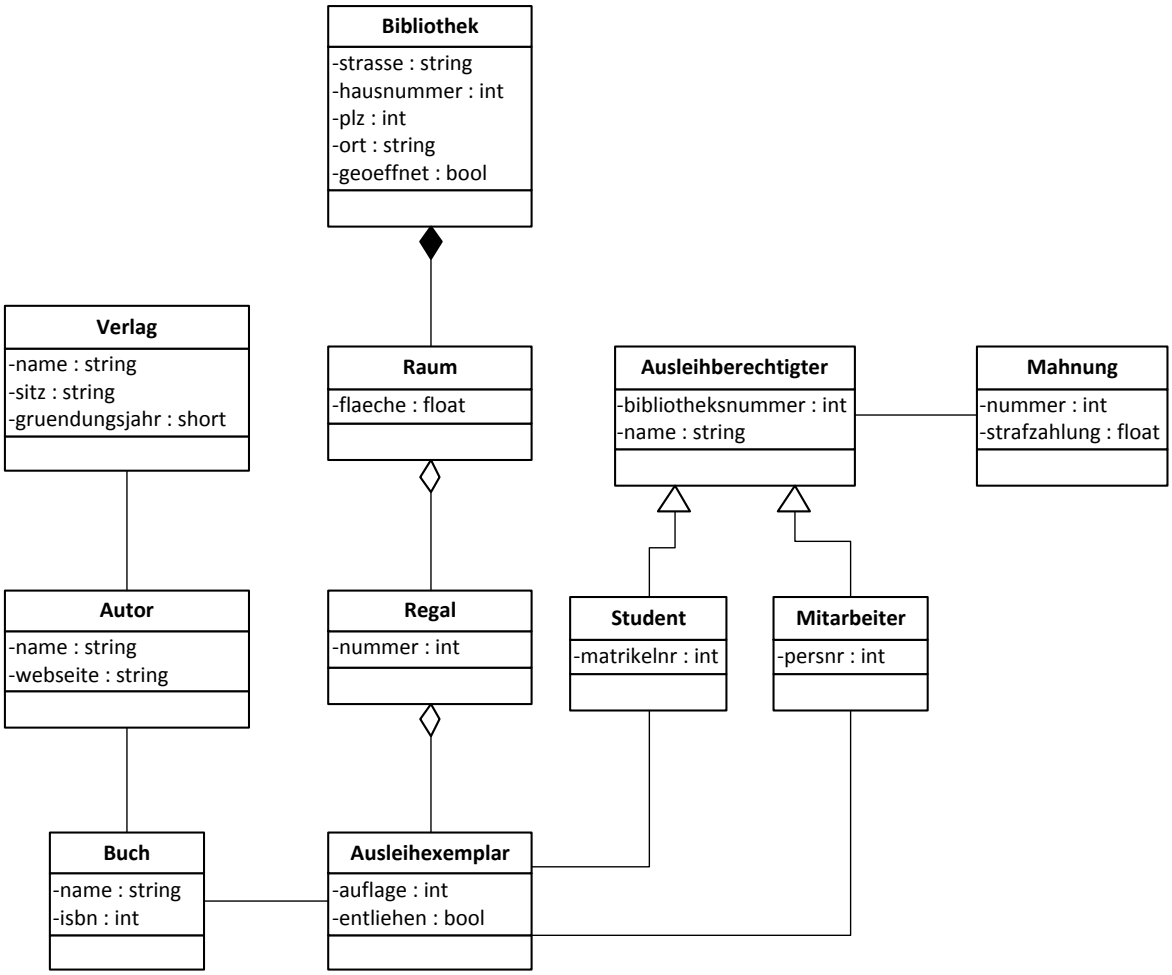

Abbildung 2: Modellierung einer Bibliothek

Lösung: Siehe Figure 4.

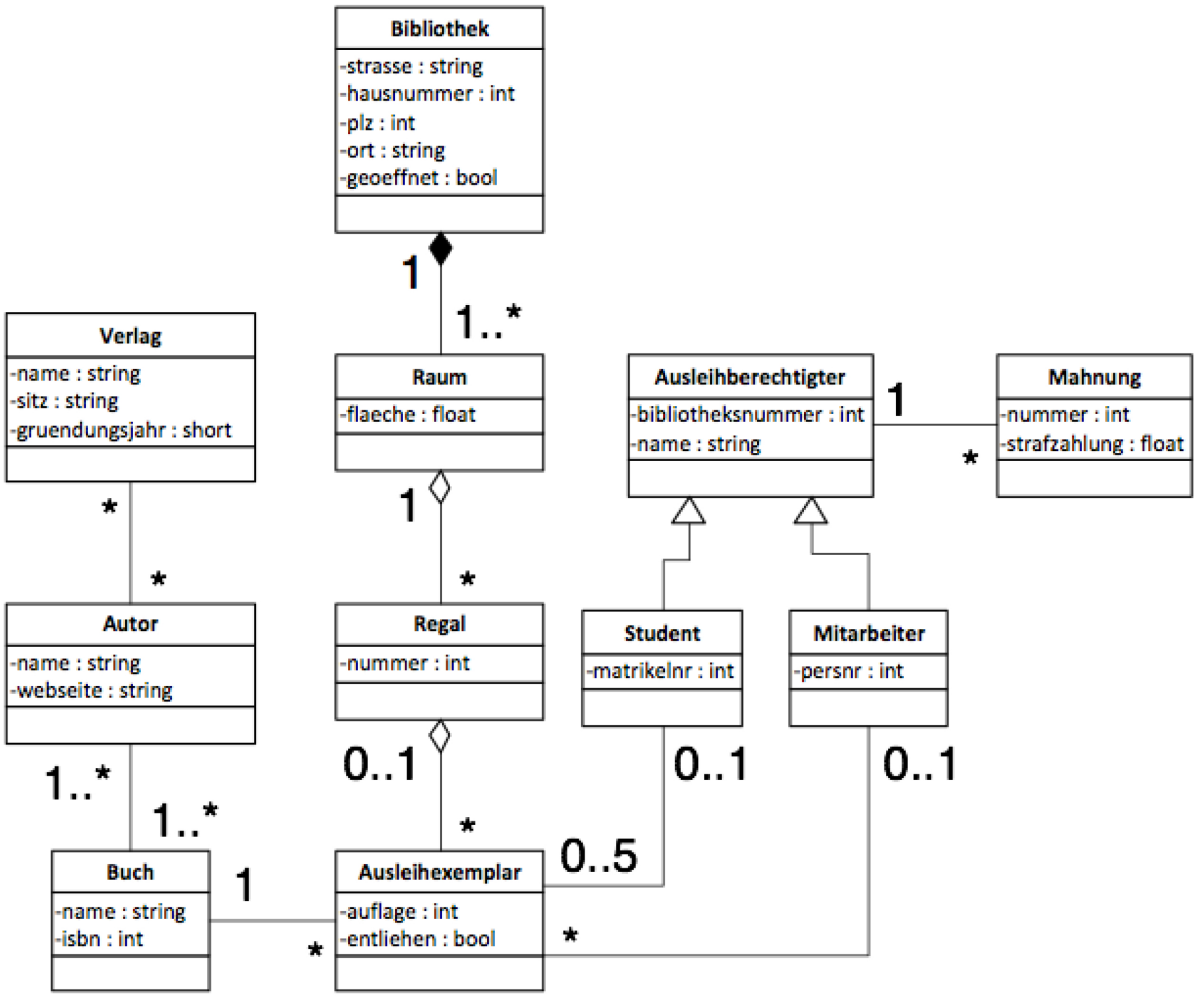

Abbildung 3: Modellierung einer Bibliothek

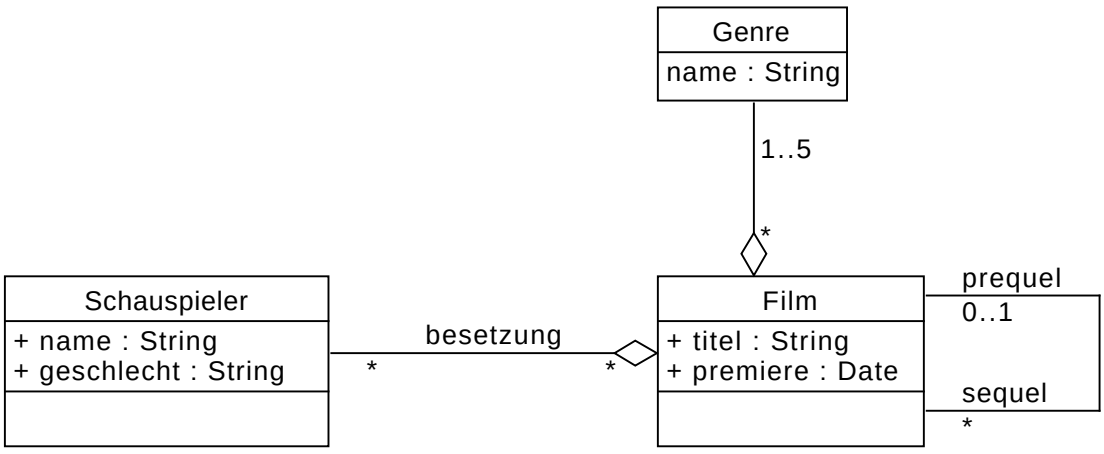

Abbildung 4: Modellierung eines Onlinestreamingservices