Furkan Aydin (1630039)

Hochschule Hannover
University of Applied Sciences and Arts

Software Engineering 1
Übungsblatt 1
UML-Klassendiagramme (Wiederholung)

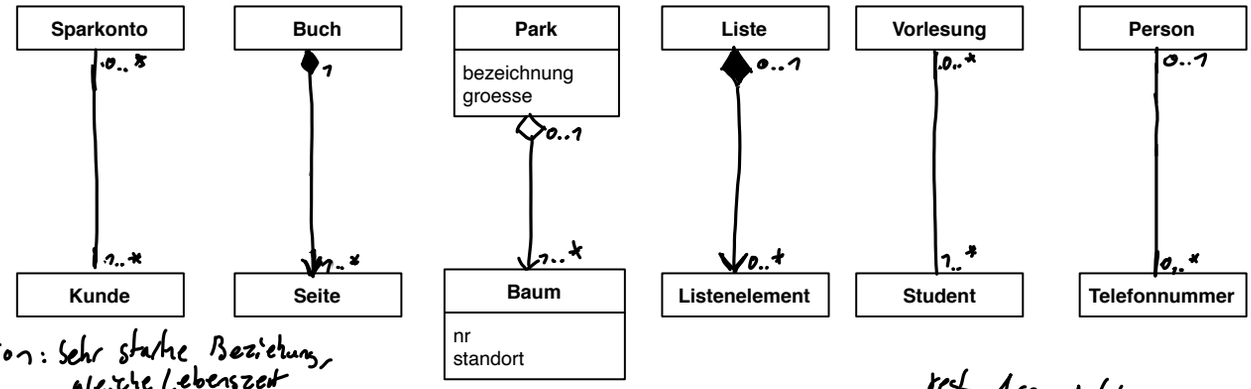
Ausgabe: 11.03.2022 **Besprechung: 17.03.2022**

Ablauf der Übungen

- Erfolgreiches Bearbeiten von 2/3 der Pflichtübungsaufgaben und persönliche Vorstellung mindestens einer Aufgabenlösung (auf Nachfrage) berechtigen zur Klausurteilnahme.
- Eine Aufgabe gilt als erfolgreich bearbeitet, wenn eine korrekte Lösung bzw. ein nachvollziehbarer Lösungsversuch termingerecht abgegeben wurde.
- Die Aufgaben sollen in 2er Gruppen bearbeitet werden.
- Die Abgabe durch **jedes Gruppenmitglied** erfolgt per Moodle vor der Besprechung in der Übung (bitte als eine PDF-Datei).
- Das Ergebnis soll in der Übung vorgestellt und diskutiert werden.

UML-Klassendiagramme wurden in Programmieren 1 und 2 eingeführt. Bitte schlagen Sie bei Bedarf in Ihren Unterlagen nach, um die nachfolgenden Aufgaben zu bearbeiten.

Aufgabe 1.1: Beziehungen zwischen Klassen (Assoziation, Aggregation, Komposition)
Bei UML-Klassendiagrammen gibt es unterschiedliche Arten von Beziehungen (Assoziation, Aggregation, Komposition). Nachfolgend sind jeweils zwei Klassen angegeben, die zueinander in Beziehung stehen sollen. Bitte entscheiden Sie jeweils welche Beziehungsart den Sachverhalt am besten abbildet, zeichnen Sie bitte das entsprechend UML-Modell mit den passenden Kardinalitäten und Navigationsrichtungen. Begründen Sie die Wahl der Beziehungsart.



Komposition: Sehr starke Beziehung, gleiche Lebenszeit
„ist ein Teil von"
„besteht aus"
Objekt ist für Existenz der Teile verantwortlich
muss bestehen

Aggregation: bindet stärker als Assoziation
„besteht ein/e"
kann bestehen

rest Assoziation

Aufgabe 1.2: Multiplizitäten / Kardinalitäten
Wieviele Objekte zu anderen Objekten in Beziehung stehen, wird in UML-Klassendiagrammen durch die Angabe von Mutiplizitäten/ Kardinalitäten angegeben.

a) Was bedeutet es, wann am Assoziationsende keine Angabe zur Multiplizität steht?
b) Was ist der Unterschied der Multiplizitäten 0..n, 0..* und * ?

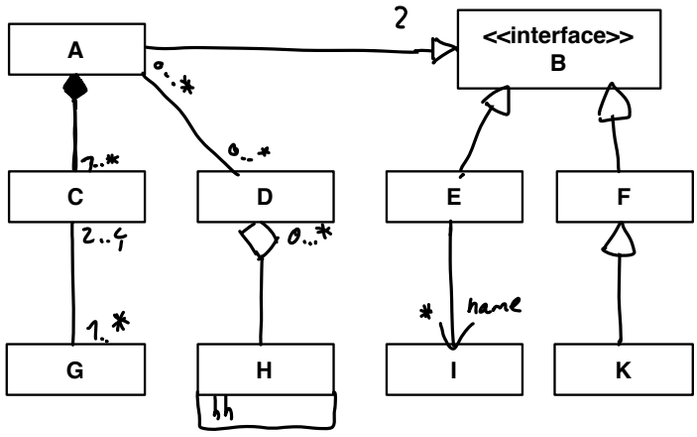
a) Dann ist die Multiplizität 1

b) 0..n ist endlich ein * entspricht 0..*
0..* ist unendlich

Aufgabe 1.3: Beziehungen zwischen Klassen

Erweitern Sie das folgende Klassendiagramm um die nachfolgenden Sachverhalte:

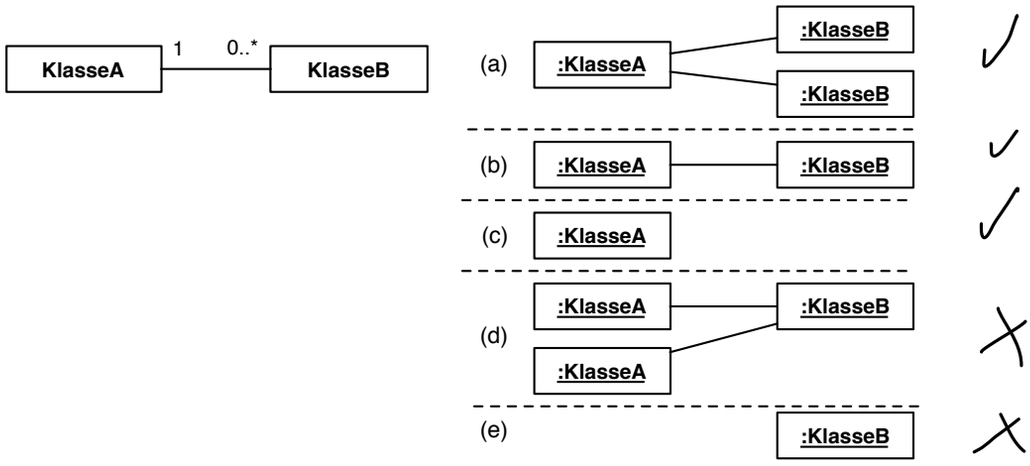
a) Die Klasse K spezialisiert die Klasse F.
b) Ein A-Objekt kennt zwei Objekte vom Typ B.
c) Beliebig viele A-Objekte kennen beliebig viele D-Objekte.
d) Jedes E kann beliebig viele I kennen, die alle ein Attribut name besitzen.
e) Beliebig viele verschiedene E-Objekte können das gleiche I-Objekt kennen.
f) Jedes H-Objekt besitzt ein Attribut hh.
g) Die Klassen E und F implementieren das Interface B.
h) Ein H kann aus beliebig vielen D bestehen.
i) Ein G-Objekt kennt mindestens 2 und höchstens 4 C-Objekte.
j) Jedes C-Objekt kann zwischen 1 und beliebig viele G-Objekte kennen.
k) Ein A-Objekt besteht aus einem oder mehreren C-Objekten, die existentiell von A abhängen.



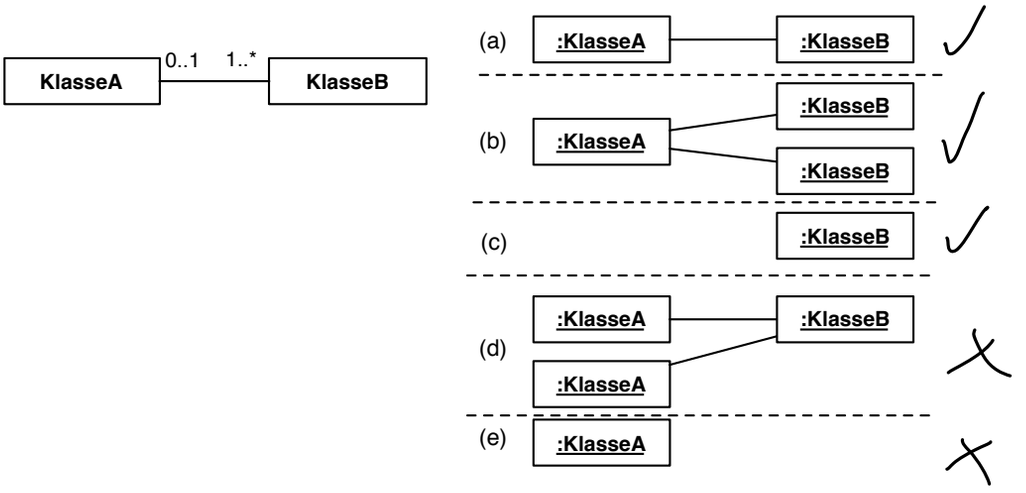
Aufgabe 1.4: Multiplizitäten / Kardinalitäten

Wieviele Objekte zu anderen Objekten in Beziehung stehen, wird in UML-Klassendiagrammen durch die Angabe von Mutiplizitäten/ Kardinalitäten angegeben. Nachfolgend sind ein Klassenmodell und zugehörige Objektkonstellationen angegeben. Bitte prüfen Sie, welche der Objektkonstellationen möglich und welche unmöglich sind.

a) Klassenmodell (a) und Objektmodelle

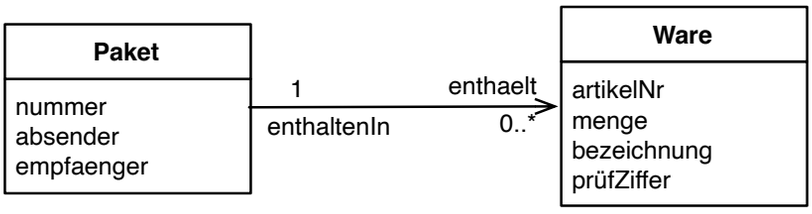


b) Klassenmodell (b) und Objektmodelle



Aufgabe 1.5: Implementierung mehrwertiger Beziehungen

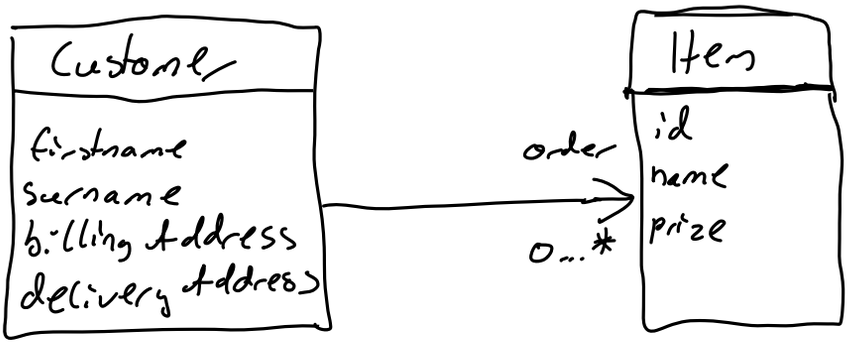
a) Skizzieren Sie die Java-Klassen (Attribute reichen), die das folgende Modell implementieren.



b) Modellieren Sie ein Klassendiagram, das die durch die Java-Klassen festgelegten Zusammenhänge darstellt.

```
public class Customer {
  private String firstname;
  private String surname;
  private Address billingAddress;
  private Address deliveryAddress;
  private ArrayList<Item> order;
}
```

b)



7.5

a)
```
public class Paket {
    private ArrayList<Ware> enthaelt;
    private int nummer;
    private String absender;
    private String empfaenger;
}

public class Ware {
    private int artikelNr;
    private int menge;
    private String bezeichnung;
    private int preisZeiter;
}
```