

Dies ist keine „Musterlösung“, sondern eine gute von vielen möglichen Lösungen. Kommentare, die nicht Teil der Lösung sind, sind kursiv gesetzt.

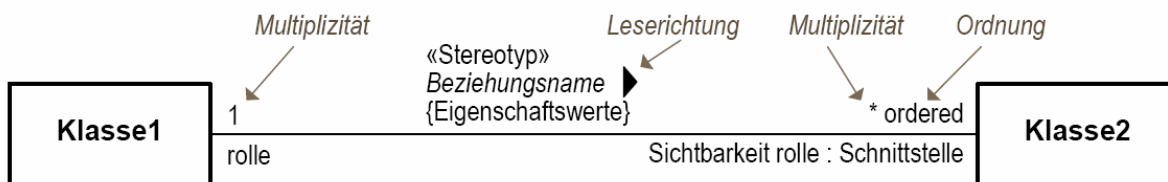
## Aufgabe 2-1\*:

Eine ähnliche Lese- und Verstehaufgabe eines Klassendiagramms gab es mal in einer SWT-Klausur.

1.a. Attribute werden in einem UML Klassendiagramm wie folgt dargestellt:



1.b. Siehe folgende Grafik:



Bei Multiplizitäten ist erlaubt:

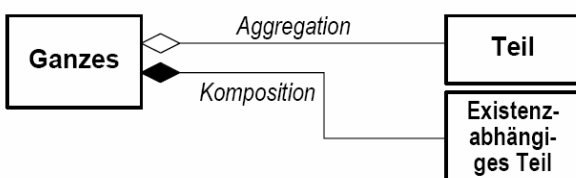
- $n..m$  mit  $n$  aus den natürlichen Zahlen und Null und  $m$  aus den natürlichen Zahlen und  $*$  und  $n \leq m$ 
  - o Achtung: Ein  $*$  steht natürlich nicht für unendlich, sondern für 0 oder mehr oder wie der Mathematiker zu sagen pflegt: beliebig, aber fest, d.h. zu einem konkreten Zeitpunkt kann ein Objekt natürlich nur mit endlich vielen anderen Objekten in einer Beziehung stehen.
- $n$  ist abkürzende Schreibweise für  $n..n$
- $*$  ist abkürzende Schreibweise für  $0..*$

Achtung: Bitte auf die Leserichtung der Multiplizitäten achten, das wird immer wieder falsch gemacht! Im obigen Beispiel gibt es zu einem Zeitpunkt zu jedem Exemplar der Klasse2 genau ein Exemplar der Klasse1 und umgekehrt zu jedem Exemplar der Klasse1 eine beliebige Anzahl (0 oder mehr) der Klasse2. Anderes Beispiel: Da ein Kind zwei Eltern hat, steht die 2 bei der Klasse der Eltern und die 1 beim Kind.

Hinweis zu den Beziehungen: Es können bei einer Beziehung zwischen Klassen (eine Unterklassenbeziehung nennt man nicht so – sie heißt in UML Vererbung) drei Namen stehen: Je ein Name an der Klasse für deren Rolle, die sie in dieser Beziehung spielt und ein Name für die Beziehung selbst (z.B. könnte eine Beziehung zwischen Klasse Person und Klasse Arzt die Beziehung „behandelt“ sein. Der Pfeil der Leserichtung zeigt dann von Arzt zu Patient. Die Person nimmt in dieser Beziehung die Rolle „Patient“ und der Arzt den „behandelnden Arzt“ ein). In der Aufgabe war letzteres gemeint. Auf dem eGK-Diagramm taucht ein solcher Beziehungsname leider nirgends auf.

Übrigens: Multiplizität wird bei Datenbanken Kardinalität genannt.

1.c. Es zeigt eine Teil-von-Beziehung mit Existenzabhängigkeit der Teile an. Schreibt man z.B. folgende Beziehung zwischen den Klassen Ganzes und Existenzabhängiges Teil, dann gilt:



Ein Objekt  $a$  der Klasse Ganzes komponiert in diesem Fall Objekte  $b_1, b_2, \dots$  der Klasse Existenzabhängiges Teil (Anzahl entsprechend der Multiplizität), d.h. wir denken uns die  $b$ s als Teile von  $a$  und die  $b$ s „sterben“, wenn  $a$  „stirbt“.

Die UML Spezifikation beschreibt dies so (UML Superstructure Specification, v2.1.1, S. 38):

*[...] the composite object (dt. das zusammengesetzte Objekt) has responsibility for the existence and storage of the composed objects (parts).*

Im Gegensatz zu den einfachen Beziehungen („Assoziation“) nennt sich dies „Komposition“. Komposition ist ein Spezialfall von „Aggregation“, welche mit der leeren Raute  $\diamond$  gekennzeichnet wird und nur die Teil-Ganzes-Beziehung darstellt (was genau der Unterschied ist, wird auf Übungsblatt 4 behandelt).

Was man genau unter „sterben“ und „Verantwortung für Existenz und Speicherung“ versteht, wenn man dann das Modell in Software umsetzt, ist erstmal nicht festgelegt. Eine Interpretation wäre z.B., dass beim Löschen von Datenbankeinträgen des Ganzen auch automatisch die Einträge für die komponierten Teile gelöscht werden.

Beispiel: Das Gehirn ist Teil (Komposition) eines Menschen (wenn der Mensch stirbt, stirbt auch das Gehirn), die Brille ist nur assoziiert mit dem Menschen (wenn der Mensch stirbt, kann die Brille mit jemand anderem assoziiert/von jemand anderem genutzt werden).

Achtung: Diese Beispiele hinken an einigen Stellen (z.B. bedeutet eine Komposition nicht, dass das Ganze „stirbt“, wenn ein Teil „stirbt“, was man im Fall Mensch<#>---Gehirn vermuten könnte).

Die Bilder sind übrigens von <http://www.oose.de/downloads/uml-notationsuebersicht.pdf>

2. Ein Rezept komponiert eine geordnete, nicht-leere Liste von Rezeptpositionen. Jede Rezeptposition ist Teil genau eines Rezepts und kann eine Dosierung als Einnahmeangabe haben.

Dass die Dosierung zwar existenz-abhängiger Teil der Rezeptposition ist, aber trotzdem Multiplizität 0..1 hat, ist wahrscheinlich KEIN Fehler (es gibt zu viele ähnliche Kompositionen in dem Diagramm), sondern entweder a.) ein Zeichen, dass die Leute der Gesundheitskarte kein UML können oder b.) sie eine uns unbekannte Definition von existenz-abhängig verwenden, bei der es durchaus Sinn macht 0..1 zu schreiben, z.B. könnte man sagen, dass die Methoden des Teils nur auf einem Objekt korrekt ausgeführt werden können, dass in Beziehungen zu einem Ganzen Objekt steht. „Existieren“ als solches in der Datenbank oder im Speicher kann es aber unabhängig.

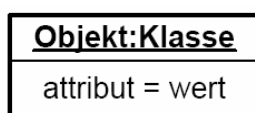
Achtung mit der Formulierung:

- „Ein Rezept besteht aus einer ... Liste von Rezeptpositionen.“ ist falsch, da ein Rezept aus noch mehr als nur der Liste besteht (nämlich z.B. einem Kommentar).
- „Ein Rezept ist eine ... Liste von Rezeptpositionen.“ ist falsch, da keine Vererbungsbeziehung vorliegt.
- „Ein Rezept hat eine Liste von Rezeptpositionen.“ ist auch nicht genau genug, da hier sowohl die Existenzabhängigkeit und die Teil-Ganzes-Beziehung nicht ersichtlich sind.

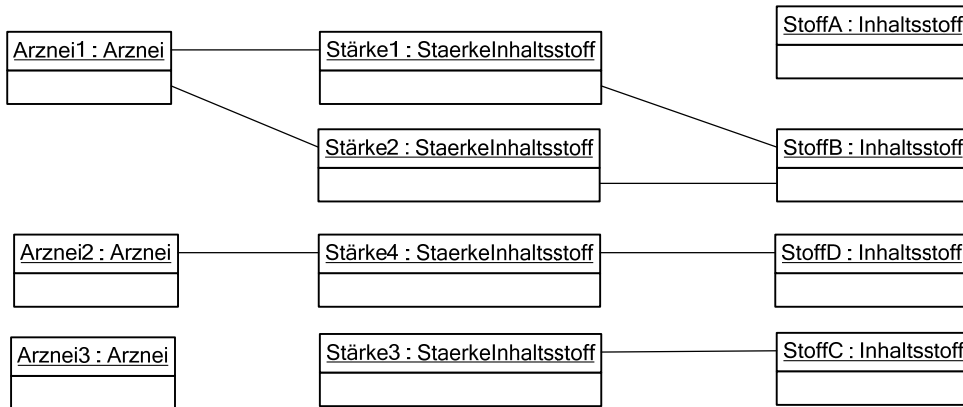
Hier als kleine Begriffsübersicht:

- Bei einer Aggregationsbeziehung zwischen A und B sagt man: A aggregiert B oder B ist Teil von A
- Bei einer Kompositionsbeziehung zwischen A und B sagt man: A komponiert B oder B ist existenzabhängiges Teil von A.
- Wir sagen:
  - A ist das Ganze oder die Komposition/ Aggregation
  - B ist das (existenzabhängige) Teil oder die (existenzabhängige) Komponente
  - Philosophisch: Da B Teil von A ist, lässt sich nicht mehr über A ohne B reden ohne sprachliche in Probleme zu geraten. In einem solchen Fall sollte man nicht von A, sondern von der Klasse A reden
- Bei einer Vererbungsbeziehung zwischen A und B sagt man: A vererbt an B oder B erbt von A
- Bei einer Assoziationsbeziehung zwischen A und B sagt man: A assoziiert B oder B steht in Beziehung zu A
- Bei benannten Beziehungen verwendet man am Besten den Beziehungsnamen inklusive Leserichtung bzw. die Rollen.

3.a. Ein Objektdiagramm stellt Ausprägungen von Klassen (Objekte) und Ausprägungen der Assoziationen zwischen diesen Klassen (Links oder dt. Objektbeziehungen) dar. Während also das Klassendiagramm beschreibt, wie Objekte potentiell untereinander zueinander in Beziehung stehen, welche Attribute diese Objekte haben und welche Operationen auf den Objekten ausgeführt werden können, beschreibt das Objektdiagramm einen tatsächlichen Zustand von konkreten Objekte, die diesem Klassendiagramm entsprechen, und deren konkrete Beziehungen zueinander zu einem Zeitpunkt. Objekte werden wie folgt dargestellt (beachte den Doppelpunkt und die Unterstreichung):



3.b. Dies zeigt die essentiellen Möglichkeiten.



**3.c.** Dass ein Inhaltsstoff in zwei verschiedenen Mengen in einer Arznei sein kann (siehe im Beispiel StoffB in Arznei1) ist sonderbar. Gibt es aber vielleicht Zwei-Phasen-Medikamentes, die den gleichen Wirkstoff in zwei Dosierungen hintereinander abgeben? Ebenfalls sonderbar ist, dass es Arzneien ohne Inhaltsstoff geben kann (siehe im Beispiel Arznei3). (Vielleicht ja Placebos? Werden die wirklich verschrieben?) Aber da uns das Bereichswissen fehlt (wir sind keine Pharmazeuten), kann man sich nicht ganz sicher sein, ob es doch Sinn macht.

*Es ist auch nicht grundsätzlich schlimm, wenn eine UML-Spezifikation etwas in der Realität nie vorkommendes zulässt. Häufig sind solche Diagramme durchaus unterspezifiziert, denn oftmals ist die Lesbarkeit auf dieser Abstraktionsebene wichtiger zu werten. Wir werden später in der Vorlesung OCL kennen lernen, mit der man genauer spezifizieren kann.*

**4.a.** Ja. Da ein BtMRezept von Rezept erbt übernimmt es auch dessen Kompositionsbeziehung zu Rezeptpositionen mit Multiplizität 1..\* und es ist damit möglich, dass ein BtMRezept drei Rezeptpositionen hat. Jeder RezeptPosition ist dann genau ein ZugeordnMedProdukt zugeordnet und diesem wiederum genau ein MedProdukt, welches dann einen Hersteller hat, da die einzelnen RezeptPositionen unabhängig sind, können sie also unterschiedliche Hersteller haben.

**4.b.** Nein, es gibt zwei Fehler:

1. „Zwei Einnahmeangaben“ geht nicht, es kann maximal eine geben.
2. Wegen Komposition und Multiplizität kann ein Dosierungsschema nur zu genau einer Dosierung gehören und nicht zu zweien.

*Achtung: Es scheint so, als hätten Einzelpositionen gar keine Dosierung, sondern nur Rezeptpositionen. (Wer auf diesen Gedanken kommt, ist schon mal zu loben!) Aber da diese Einzelpositionen sind, \*gibt\* es also Einzelpositionen, die Dosierungen haben – wenn auch wohl nicht alle. Aber: Wegen des kursiven Namens ist Einzelposition \*abstrakt\*, d.h. Einzelposition kann nicht ausgeprägt werden bzw. es gibt keine Objekte, die Einzelpositionen sind, aber nicht eine ihrer Unterklassen. Da es keine anderen Unterklassen gibt, sind also im Rahmen dieses Diagramms (closed world assumption) alle Einzelpositionen Rezeptpositionen und haben somit eine Dosierung.*

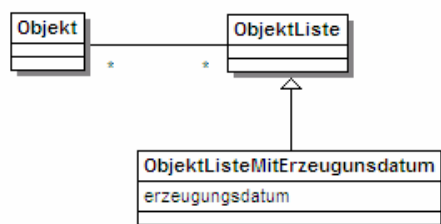
*Achtung 2: Der zweite Fehler gilt, da vom **selben** Dossierschema die Rede war. Dies ist wegen der Multiplizität nicht erlaubt (jedes Dossierschema hat genau eine Dosierung). Zwei **gleiche** Dossierschema wäre aber gegangen (im Sinne von: Die Attribute der Dossierschemen haben die gleichen Werte).*

## Aufgabe 2-2:

**1.a.** Eine Objektliste-Mit-Irgendwas \*ist\* eine Objektliste, sie enthält (=aggregiert) sie nicht. Könnte sein, dass man es so implementiert, weil es aus irgendeinem Grund sinnvoll ist (z.B. Effizienz), man \*modelliert\* es aber niemals so, denn es entspricht nicht der Realität.

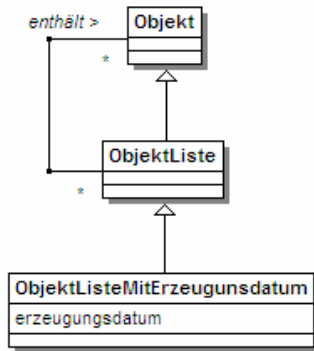
*Achtung: Mehrmals wurde in euren Lösungen geschrieben „Nein das geht nicht, weil Objekte nicht in einem Klassendiagramm auftauchen dürfen“. Das ist natürlich Unsinn, denn obwohl wir in der Tat keine Objekte in einem Klassendiagramm haben dürfen, dürfen wir doch die Klasse der Objekte haben.*

Eine Objektliste-Mit-Erzeugungsdatum würden wir also wahrscheinlich so modellieren:



(Die Multiplizität ist auf beiden Seiten \*, da ein Objekt ja in vielen Listen enthalten sein kann)

Wenn man ganz genau sein wollte, kann man auch noch die Vererbungsbeziehung von Objekt zu ObjektListe einzeichnen (da ja alle Klassen von Objekt erben):



**1.b.** Hier andersherum: Ein Fernseher ist nicht eine Röhre, er *\*enthält\** eine Röhre. Außerdem sind Empfangsgerät und Gehäuse nicht komponiert, sondern können auch unabhängig vom Fernseher existieren. *D.h. sie sind lediglich aggregiert (wieder unter Berücksichtigung, dass man immer auch eine andere Definition von existenzabhängig anlegen könnte, aber die verwendete passt für echte Dinge ganz gut*

**1.c.** Ein B kann nicht in 2 As komponiert sein.

2. Der Begriff Methode ist wahrscheinlich als Alternative zu Prozedur im objektorientierten Kontext entstanden. Methode suggeriert die Regelmäßigkeit der ausgeführten Anweisungen: Alle Objekte einer Klasse „arbeiten nach der selben Methode“ (im Sinne von Arbeitsweise).

## Aufgabe 2-3:

Jeder hat eigene Vorstellungen. In den meisten InformatikerInnenjobs geht es aber in irgendeiner Weise um die Erstellung von Software und damit automatisch um Softwaretechnik. Es kann aber sein, dass grad der erste Job sehr kodiernah ist und die „Kunden“ selbst Programmierer sind und somit ALP relevanter ist. Mit fortschreitender Karriere und zunehmenden Offshoring (= Verlagerung von Jobs in Billiglohnländer) wird sich dies aber sicher ändern!