

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

Aufgabe 4-1:

1. Aggregation und Komposition sind beides Besitz- („hat ein“-) Beziehungen zwischen Objekten. Bei einer Aggregation ist das enthaltene Objekt eigenständig sinnvoll, z.B. hat ein Mensch eine Brille; die wird z.B. abgenommen und (ohne den Menschen dabei haben zu müssen) repariert. Bei einer Komposition hingegen ist das Teil fest mit seinem „Wirt“ verbunden, z.B. das Gehirn bei einem Menschen. Das Gehirn stirbt mit dem Menschen. Es ist von dem Problembereich abhängig, ob etwas als Komposition oder als Aggregation modelliert wird, je nach den obigen Kriterien. *Java unterscheidet zwischen diesen beiden nicht, C++ sehr wohl. Aber Programmiersprachen sind an dieser Stelle eigentlich nicht das richtige Denkmodell, wir sind noch weit vom Programmieren weg. Bitte also nie mit Programmiersprachen argumentieren! Und Achtung: Eine Komposition ist in der realen Welt eher selten. Im Zweifelsfalle immer eine Aggregation nehmen.*

Ein Attribut hingegen ist gar kein Objekt (keine Objektidentität), sondern nur eine Eigenschaft (meist ein Skalar) eines Objektes und als solche fest mit ihm verbunden. Eigenschaft eines Menschen ist z.B. seine Haarfarbe oder sein Name.

Attribute werden (wie die Operatoren) in dem Klassenviereck angegeben, Aggregationen und Kompositionen mittels einer Verbindungsgerade gezeichnet, mit einer Raute auf Seite des Besitzers. Bei Kompositionen ist diese Raute ausgefüllt, bei Aggregationen leer.

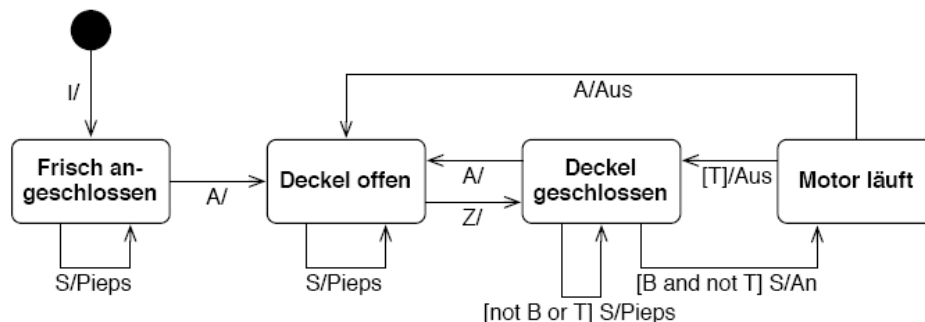
2. Aktionen (action), die in den Zuständen angegeben werden, werden ausgeführt wenn dieser Zustand erreicht (entry) oder verlassen (exit) wird. Aktionen an Übergängen hingegen werden ausgeführt bei genau diesem Übergang, also noch vor einer entry-Aktion des resultierenden Zustands. Aktionen sind „unmittelbar“ und ohne zeitliche Ausdehnung. Aktivitäten (do) hingegen haben eine zeitliche Ausdehnung und können zudem nur mit einem Zustand assoziiert werden. Siehe Bild:



3. Aktivitätsdiagramme verlangen keine Angabe von Objekten/Klassen. Mit ihnen lassen sich auch besser nebenläufige Prozesse beschreiben – insbesondere deren Synchronisation (*Petrinetz-Ähnlichkeit, für die die verstehen was gemeint ist...*). Ein weiterer Vorteil ist die übersichtlichere Darstellung von Alternativen/Entscheidungen. Es ist auch mehr Daten-/Objektfluss darstellbar, was ein Diagramm aber andererseits recht kompliziert werden lassen kann, wenn man es mit dem Kontrollfluss mischt. Bei Sequenzdiagrammen ist Kontrollfluss gleich Datenfluss.

4. Keine, sie gleichen denen der Menschen als Akteure: Der Akteur als System muss außerhalb des zu bauenden Systems liegen, insbesondere haben wir keinen Einfluss auf dieses System. *Dies wird häufig falsch gemacht.*

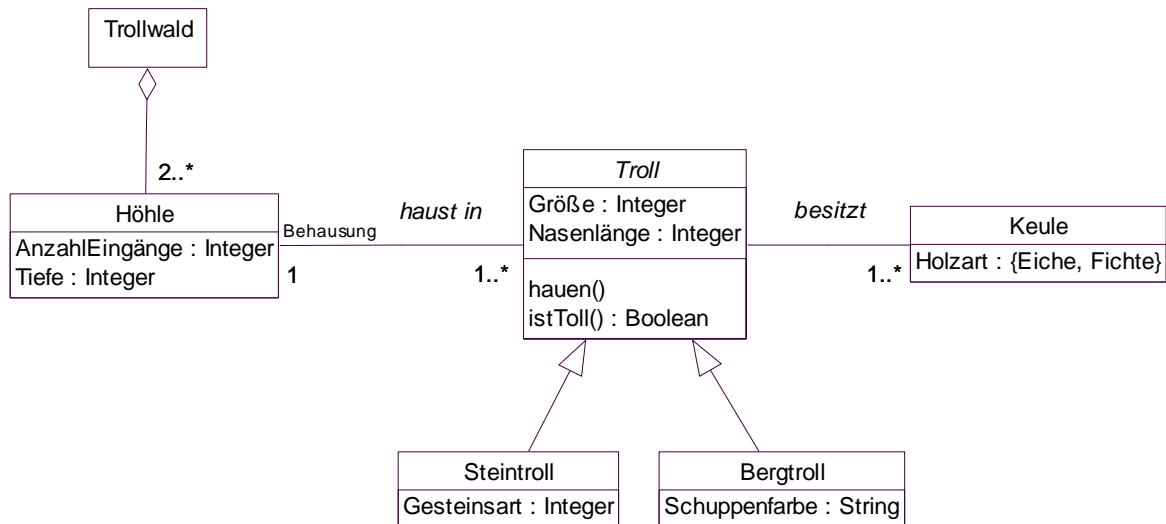
Aufgabe 4-2: Diese Aufgabe war mal eine Klausuraufgabe.



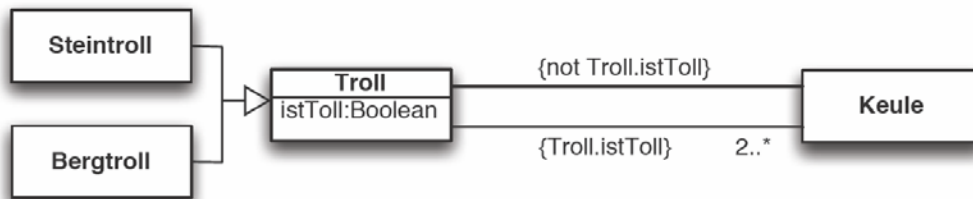
Im Zustand 1 wird davon ausgegangen, dass der Deckel geschlossen ist.

Eine echte Schleuder wird übrigens niemals Ereignissensoren wie „auf machen“ und „zu machen“ haben (wie soll man das technisch machen?), sondern es können höchstens „offen“ und „geschlossen“-Zustände vorliegen.

Aufgabe 4-3:



Mit Ausnahme der Holz-Typisierung, müssen die anderen Attribute keinen Typ haben, da er ja nicht klar angegeben wurde. Merke: Im Zweifelsfall unspezifiziert lassen. Troll ist abstrakt (der Name ist kursiv), da es nur Stein- oder Bergtrolle gibt. (Auch das ist aber nicht klar gesagt.) Die besitzt-Relation könnte auch als Aggregation ausgeführt werden. Im Zweifelsfall nimmt man aber immer eine einfache Beziehung. Ähnliches gilt für die Beziehung Trollwald und Höhle. Was ist mit „Toller Troll“? Man könnte es weglassen (schlecht), man kann es aber auch als Attribut in Troll rein nehmen oder wie hier als Methode „istToll():Boolean“, denn es ist vielleicht berechenbar, z.B. aufgrund der Anzahl der Keulen. Leider fehlt bei allem die Verbindung zwischen Anzahl Keulen und Tollsein. Man könnte es wie folgt machen:



In {} stehen Bedingungen für Beziehungen, sogenannte Diskriminatoren. Leider ist hier die Wirkungsrichtung falsch: Aus den Keulen muss sich die Tollheit ergeben, nicht wie hier andersherum. Richtig gut kann man das erst mit OCL modellieren, aber das kommt erst später in der Vorlesung. Ähnliches gilt für die im Text erwähnte Unterscheidbarkeit der Trolle anhand ihrer Attribute.

Aufgabe 4-4:

Eine mögliche Meinung: Architekturmodelle (Komponentendiagramme oder Paketdiagramme) und Simple Zustandsdiagramme und Sequenzdiagramme werden vom Kunden verstanden, mehr nicht, d.h. der Rest sollte Text sein. Im Allgemeinen sind Klassendiagramme weder leicht verständlich, noch deren Sinn klar. Es sollte sich immer etwas bewegen: dynamische Diagramme ist das, was der Kunde will!