

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

**Aufgabe 6-1:**

Die Beispiele stammen aus dem eRezept-Transport von Aufgabe 3-2.3

a.

**Bereichsklassen** sind Klassen aus dem Anwendungsbereich und somit die Sprache der Anforderungsanalyse.

**Lösungsklassen** sind programmiertechnische Umsetzungen (die nicht zwingend 1:1 sind) und sind erst beim Entwurf relevant.

**Beispiel:** eR-Ticket als Bereichsklasse bekommt als zentrale Datenstruktur im Entwurf eine HashMap als Lösungsklasse.

b.

**Geschäftsobjekte** (dieser Begriff hat sich statt Geschäftsklassen im Deutschen durchgesetzt) beschreiben Daten-beinhaltennde Klassen, meist Abbildungen realer Entitäten aus der Anwendungswelt. Beispiel: eRezept.

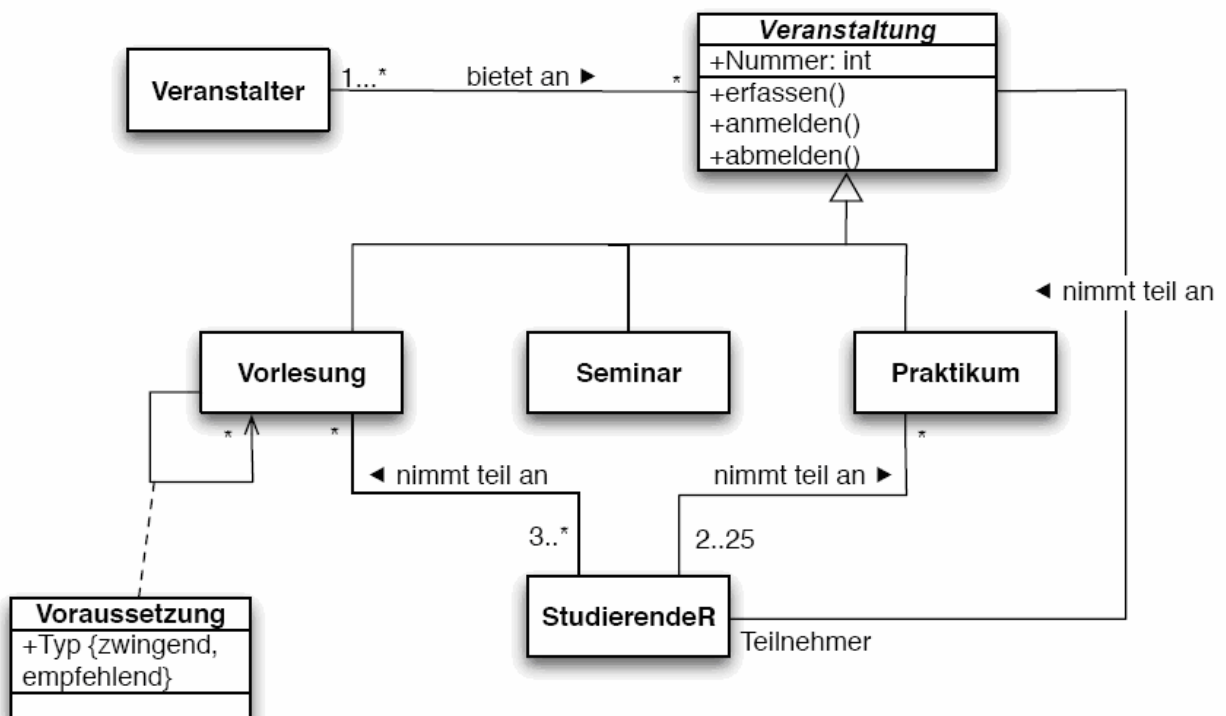
**Zugangsklassen** sind die Klassen, mit denen Akteure in der Regel Zugriff auf das System bekommen. Das sind nicht nur Klassen aus der Benutzungsoberfläche, sondern auch API-Klassen oder allg. Schnittstellenklassen, die z.B. den Zugriff auf die interessanten Geschäftsobjekte ermöglichen. Beispiel: Fällt leider schwer hier, da keine konkreten Klassen angegeben werden, aber die eGK selbst hat ein Stellvertreterobjekt zum Schreiben auf die Karte; dies ist eine reine Schnittstellenklasse, also eine Zugangsklasse.

**Steuerklassen** sind vor allem durch ihren dynamischen Anteil wichtig: Sie sorgen für den zentralen Ablauf einer Operation. In einem Sequenzdiagramm gehen z.B. die meisten Pfeile von ihnen aus. Sie sind meist im System verborgen.

Beispiel: PVS oder AVS.

**Aufgabe 6-2:**

1.a. Hinweise: Veranstaltung ist abstrakt, deshalb der kursiv gesetzte Name. „nimmt teil an“ wird in den Unterklassen überschrieben. Die Assoziationsklasse Voraussetzung gehört zu einer reflexiven Assoziation von Vorlesung. Man beachte auch die Richtungspfeile bei „bietet an“ und „nimmt teil an“. Letzteres muss in beiden Richtungen navigierbar sein; das ist was anderes als die Leserichtung! Teilnehmer ist die Rolle von StudierendeR in einer Veranstaltung.



1.b.

Viele **Substantive** sind hier eindeutig die Klassen, man muss allerdings Singular und Plural auseinander halten. *Klassennamen sind immer im Singular zu halten!* Es gibt Substantive, die nicht Klassen sind, nämlich die Phasen und Zeitangaben, wozu auch „Prüfung“ gezählt wurde. Dies sind Ereignis-Substantive. Teilnehmer ist eine Rolle von Studierenden in einer Veranstaltung. Die „Reihe“ ist keine Bereichsklasse, sondern allenfalls eine Lösungsklasse und hat also hier nichts zu suchen. Genauso wie die Pluralangabe („Teilnehmer“) drückt sich dies in der Multiplizität der Assoziationen aus. Das „ist“ in b) wurde zur Vererbung, das „sind“ in c) aber nicht.

**Attribute:** Nur „Nummer“ und der von Abbott kaum zu entdeckende „Typ“ in Voraussetzung.

**Operationen:** Die Operationen sind gemäß Abbott klar erkennbar. Methoden von Vorlesung wurden in die Oberklasse geschoben.

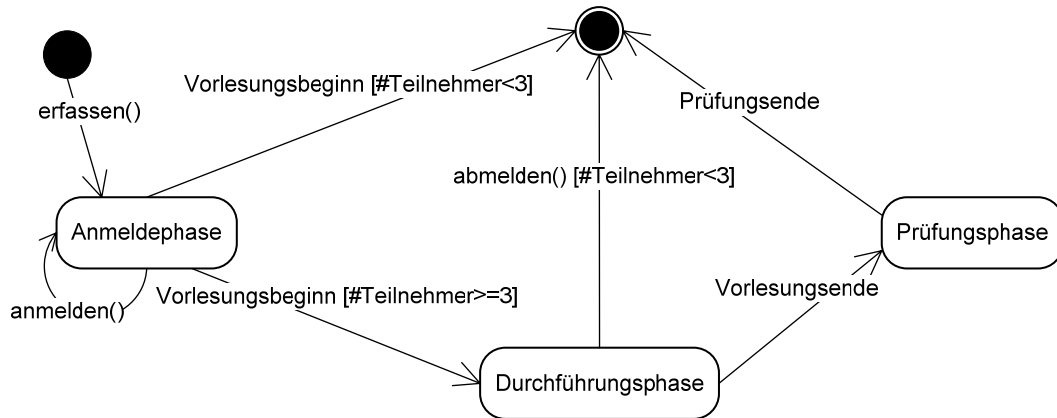
**Assoziationen:** „Voraussetzung“ ist eine Assoziation, hier als Assoziationsklasse. Abbott hilft hier wenig. Keine der „hat“ wurde zur tatsächlich Aggregation: In e ist es die Umkehrung von „nimmt teil an“, in f die Beziehung zu den Zuständen eines Objekts (etwas was Abbott gar nicht thematisiert).

**Multiplizitäten/Constraints/Rollen:** S.o. bei Substantiven.

**Fazit:** Es hilft nur oberflächlich und meist gibt es genauso viele Ausnahmen und Uneindeutigkeiten wie klare Fälle. Aber es bewahrt vor dem Leeren-Blatt-Syndrom.

1.c. Grundlage ist der Lösungshinweis zu 5-2: erfassen() entspricht „Veranstaltung eingeben“ und anmelden() entspricht „Vorlesungstermin auswählen“. Man müsste hier eine Vereinheitlichung der Terminologie beginnen und die Diagramme anpassen.

2. X-beginn und X-ende sind Zeitereignisse, die anderen sind Operationen des Vorlesungsobjektes. anmelden() wurde der Vollständigkeit wegen eingeführt. #Teilnehmer heißt „Anzahl Teilnehmer“ und bezieht sich auf die Assoziation „nimmt teil an“. *Gibt es dafür eine spezielle Notation in UML (z.B. count(...)?*



**Aufgabe 6-3:**

Das Vorgehen ist zunächst sehr Objekt- und Klassenorientiert. Damit werden zum einen Datenrepräsentationen stark repräsentiert, zum anderen Interaktionen zwischen den Daten beinhaltenden Objekten. Dies führt meist zu einer datenflussorientierten, Objektnetz-ähnlichen oder ablagebasierten Architektur (siehe Architekturstile später). Dies sind typischerweise Informationssysteme. Weniger typisch ist dies für Realzeitsysteme (mit hohem Kontrollflussanteil) oder eingebettete Systeme (wo es viele externe Vorgaben gibt).