

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

Aufgabe 3-1:

1. Eine Klasse beschreibt Objekte gleicher Art und ist keine Menge. Bei einer Menge müssen die Objekte nicht einmal aus der gleichen Klasse kommen.

2.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.*

2.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 – aber das kommt später in der Vorlesung.*

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

3. Eine Methode beschreibt eine Methode (= Art und Weise), wie eine Operation zu realisieren ist. Stichwort: Polymorphismus/Überladung/Vererbung der Schnittstellen. Beispiel: Die Operation `zeichne()` gibt es für `GeometrischesObjekt`, wovon `Kreis` und `Rechteck` Unterklassen sind. Diese haben beide ihre eigene „Methode“, wie sie zu zeichnen sind – welche offensichtlich voneinander verschieden sind. *Wegen der Verwechslungsmöglichkeit mit einer wiss. oder softwaretechn. Methode sollte man besser „Operation“ sagen. So heißt es auch in UML.*

Aufgabe 3-2:

1. Strukturdiagramme stellen statische Modelle dar: Unveränderliche Eigenschaften und Strukturen des Systems und der Daten. Verhaltensdiagramme stellen dynamische Modelle dar: Zeitliche Abläufe der Interaktionen zwischen Teilsystemen und Objekten und des Lebenszyklus eines einzelnen Objekts.

2.a. Klassen (als Typangabe der Objekte, nicht als Graphik) und Methoden.

2.b. Die Zeit oder die Lebenslinie

2.c. *Das wird oft vergessen!* „Oben“ stehen Objekte, keine Klassen. Wenn von konkreten Objekten die Rede ist, dann kann also das Sequenzdiagramm auch nur einen konkreten (von vielen möglichen) Ablauf der Interaktion dieser Objekte darstellen. Es ist also nur ein Beispiel, wie es abläuft. So wird z.B. nicht angegeben, was passiert, wenn eine Ausnahme geschieht; man nimmt für diesen Fall/Beispiel an, dass sie nicht geschieht. Sequenzdiagramme sind also **nicht** äquivalent zu programmierten Methodenrumpfen

3. Siehe nächste Seite. *Diese Lösung nicht eindeutig, da die statische Modellierung nur geraten werden kann. In der vorliegenden Lösung kann z.B. das eRezept sich selbst verschlüsseln (enc). Das muss nicht so sein. In der Vorlesung hat Herr Prechelt gesagt, dass das authC2C() auch die Karte selbst (mit dem HBA) machen könnte, da sie einen Prozessor hat. Das HBA ist hier Teil des PVS geworden; es kann aber auch ein eigenständiges Objekt sein. Dann sieht's anders aus. Die Erzeugung vom Ticket und vom eRezept ist nicht mit aufgenommen, auch das könnte man machen. Bitte ausprobieren!*

Aufgabe 3-3:

Der Entwickler sagt:

* Nur ich kann die speziell gebrauchte Funktionalität realisieren. Bei Fremdem weiß ich nicht, ob es wirklich das tut, was ich erwarte. („Does-Not-Apply-Here-Syndrom“)

* Ich will mich nicht abhängig machen von dem Hersteller. Wenn etwas nicht funktioniert, kann ich nichts machen, bekomme aber den Ärger. Außerdem kann es sein, dass sich die Funktionalität in neuen Versionen ändert – das muss ich jedes Mal prüfen.

* Ich muss erst diese andere Bibliothek finden. Das Suchen ist aber nicht so einfach. Außerdem muss man Evaluieren, welche der vielleicht vorhandenen Alternativen die geeignetste ist – das ist so „unkreativ“.

* Einarbeiten in fremde Bibliotheken macht weniger Spaß als selbst programmieren. („Not-Invented-Here-Syndrom“)

Hier könnte man gut fragen, wie diese Probleme beseitigt werden könnten.

Der Manager sagt hingegen:

* Eine Komponente kaufen ist häufig billiger und oft schneller als Selbstmachen.

* Aber Abhängigkeit ist in der Tat nicht schön, vor allem kann der Hersteller pleite gehen oder die Unterstützung für das Produkt einstellen.

* Wenn der fremde Hersteller einen guten Namen hat, wertet es unser Produkt auf. Und gute Geschäftsbeziehungen sind immer hilfreich.

* Die Wartung und Gewährleistung für dieses Modul liegt nicht mehr bei uns (Outsourcing).

