

Neue Vorgaben für das Zentralabitur 2012: Entwurfs- und Implementationsdiagramme

Rückschau:

Die bisherige Notation der Klassendiagramme waren eine (nahezu) äquivalente Darstellung lauffähiger Programme ohne Methodeninhalt. Mit geeigneten Entwurfswerkzeugen (z.B. UMLed) konnte man sogar syntaktisch korrekten Quelltext erzeugen. Die Darstellungen wurden dadurch häufig sehr umfangreich, komplex und aufwändig in der Darstellung.

Methodische Konsequenzen:

- Im praktischen Unterrichtseinsatz kam es oft zu „irgendwie“ verkürzten Darstellungen im Heft oder an der Tafel. Eine vollständige Darstellung war gerade *während* der Modellierung umständlich.
- „Schwache“ Schüler verstanden die Diagramme nicht, da sie beim Blick auf die oft komplexe und überladene Darstellung die zugrunde liegende Idee nicht abstrahieren konnten.
- „Starke“ Schüler neigten dazu, die Entwurfsphase zu vernachlässigen und waren mit dem Kopf oft schon beim Programmtext, der Algorithmisierung, dem Entwurf der GUI etc..

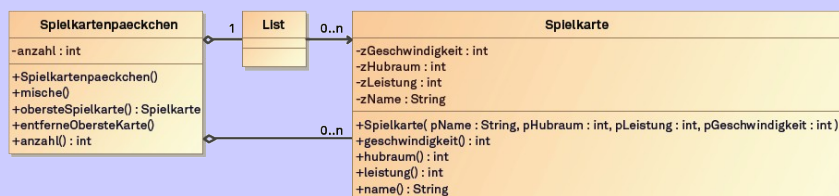
Didaktische Konsequenzen:

- Die Phasen der Planung, Modellierung und Programmierung verschwammen, Schüler mit Stärken in der Programmierung hatten Vorteile, Schüler mit Stärken in der Modellierung konnten sich nicht ausreichend profilieren.
- Die programmiersprachenspezifischen Aspekte waren den davon unabhängigen Aspekten deutlich überbetont.

Außerdem:

- Die bisherige Darstellung hatte inhaltliche Schwächen (siehe Beispiel)
- Die bisherige Darstellung hat die Komposition als Objektbeziehung nicht ganz korrekt verwendet.

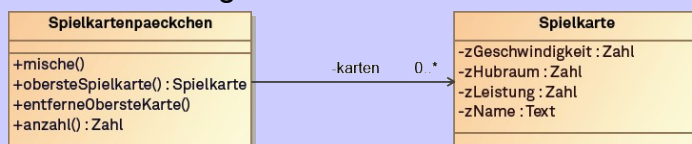
Klassendiagramm



- hier: überflüssige Komposition (Hat-Bez.)
- Komposition nicht genau angewendet
- (fast) alle Methoden und Attribute
- sprachspezifische Typbezeichnungen
- Vermischung von Modell (Analyse, Entwurf) und Programm (Implementierung)

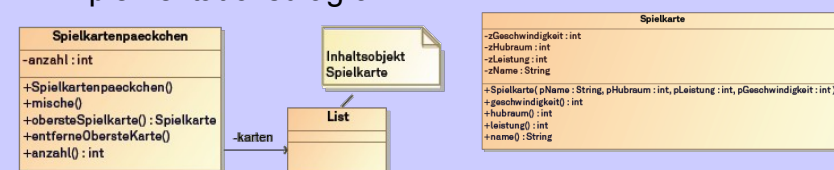
wird ersetzt durch:

1. Entwurfsdiagramm



- reine Modellsicht
- plausiblere Darstellung bzgl. der Domäne und der Multiplizitäten
- verkürzte Darstellung
- Datenstrukturen als implementations-elemente entfallen
- Beziehungen können bezeichnet werden

2. Implementationsdiagramm



- vornehmlich technische Sicht
- Kommentare möglich
- Datenstrukturen können korrekt mit beliebigen Inhaltstypen (-klassen) dargestellt werden
- Komposition (Hat-Bez.) kann entfallen

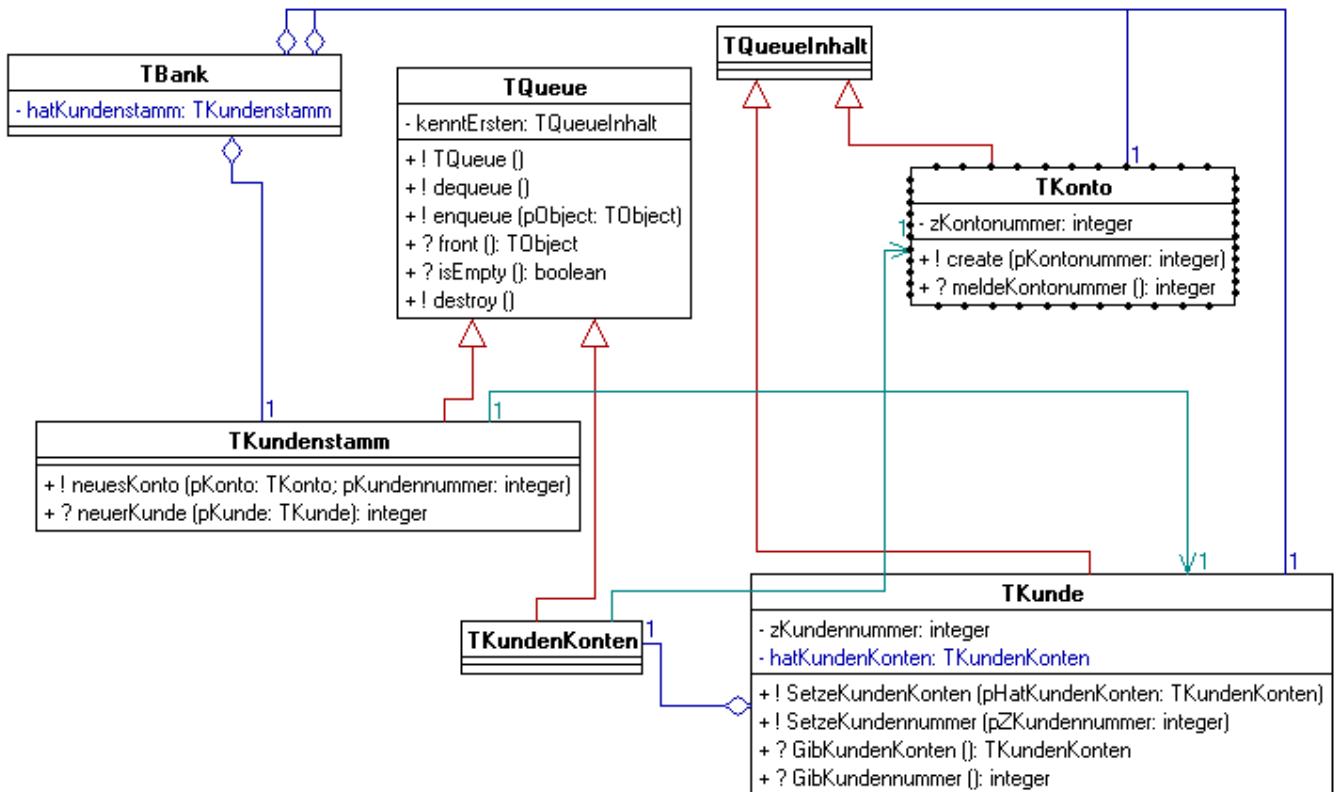
Ziele der Umsetzung von Entwurfs- und Implementationsdiagrammen:

- Schwächen i.d. Darstellung ausmerzen (z.B. überflüssige Hat-Beziehungen)
- Deutlichere Trennung von Modellierung und Implementierung
- Bessere, weil sinnvoll verkürzte und damit übersichtlichere Darstellung in Analyse- und Entwurfsphase

Beispiel

In einem Diagramm soll eine Bank modelliert werden, in der die Kunden jeweils mehrere Konten besitzen soll.

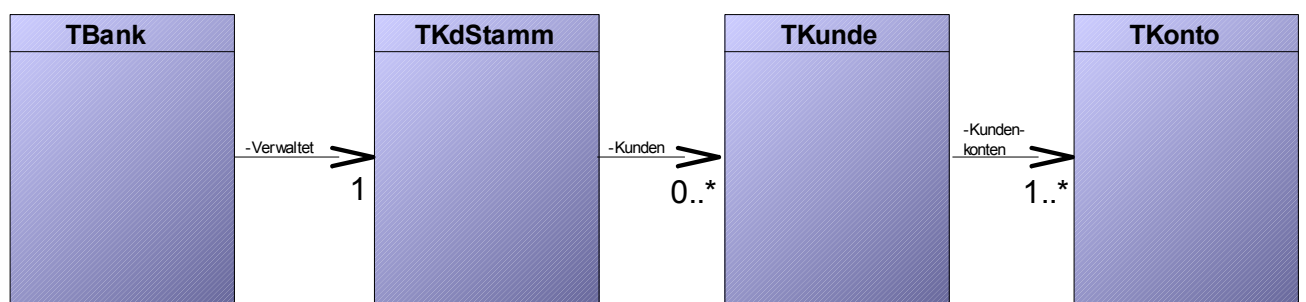
Mögliche bisherige Darstellung:



Dieses Modell ist längst nicht vollständig, zeigt aber eine Vielzahl von Objektbeziehungen, die rein programmieretechnisch bedingt sind. So unterhält das Objekt der Klasse TBank die beiden Hat-Beziehungen zu den Klassen TKonto und TKunde, nur um neue Kunden anzulegen oder ihnen neue Konten zuzuordnen.

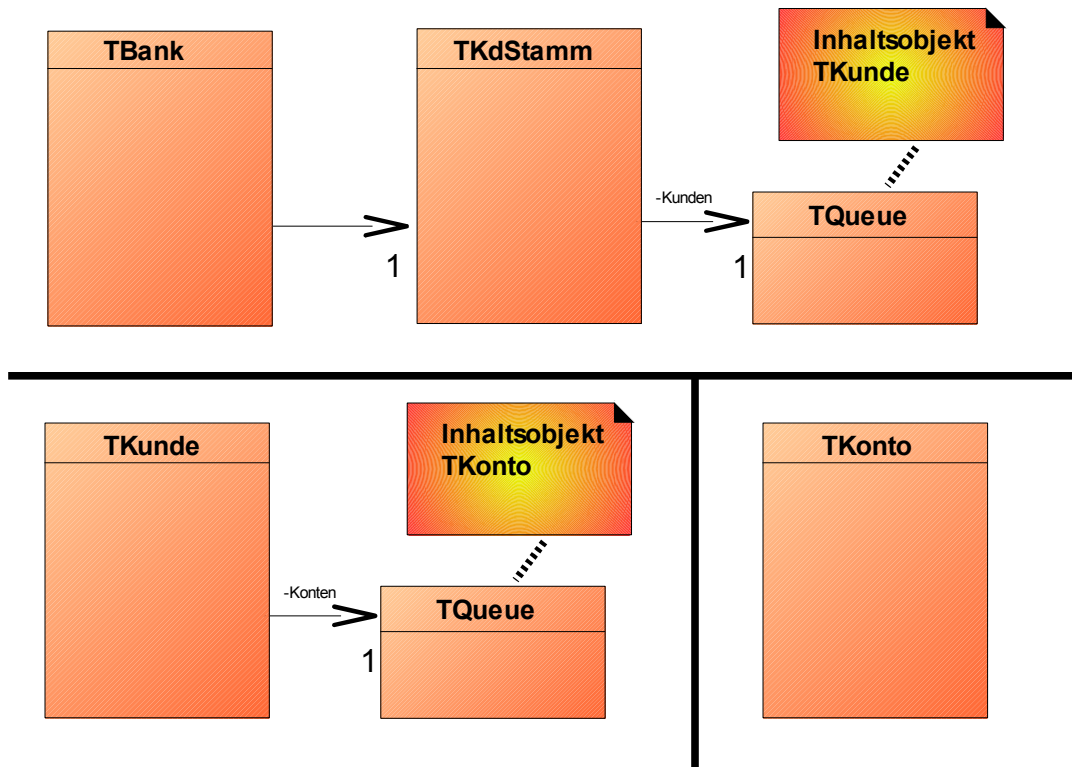
Hier zeigt sich auch die an sich falsche Anwendung der Hat-Beziehung (Komposition). Diese beschreibt eigentlich ein weitreichendes Anhängigkeitsverhältnis: Wird das Objekt der komponierenden (besitzenden) Klasse gelöscht, so müssten eigentlich alle komponentierten (besessenen) Objekte auch gelöscht werden. Dies ist bei uns nicht der Fall, da Objekte solange „leben“, wie sie durch eine Komposition oder mindestens eine Assoziation referenziert werden.

Eine viel intuitivere und an die Domäne angepasstere Sicht:
(Darstellung unvollständig)



Streng genommen gilt die speichernde Datenstruktur als technisches Detail. Es spielt zur Entwurfszeit noch keine Rolle, ob es sich um einen Stack, einen Queue etc. handelt. Dies wird dann im Implementationsdiagramm konkretisiert:

(Darstellung ohne Attribute und Methoden)



Entwurfsdiagramme:	Implementationsdiagramme
- vereinfachte Darstellung - keine Konstruktoren und Selektoren - Datenstrukturen spielen i.d.R. Noch keine Rolle - Beziehungen werden mit Bezeichnungen und (einseitigen ¹) Multiplizitoren versehen - Datentypen (Zahl, Zeichenkette, Wahrheitswert) sind sprachenunabhängig	- Methoden und Attribute werden mit vollst. Deklaration angegeben - Datenstrukturen werden angegeben, die konkreten Inhaltsobjekte jedoch nur als Kommentar – sie werden getrennt beschrieben.

Hinweise:

Es handelt sich hier ausschließlich um eigene Überlegungen zur Umsetzung der Entwurfs- und Implementationsdiagramme in meiner Fachschaft. Sie entstanden nach Gesprächen mit Klaus Dingemann und Prof. Dr. Jan Vahrenhold, sind aber durch keinen der beiden legitimiert.

Nachweise:

- Präsentation zum Vortrag von Jan Vahrenhold (Fachworkshop Informatik zur Fachtagung Zentralabitur 2012)
- Harald Störrle: UML 2 für Studenten (Kapitel 5: Klassen und Beziehungen)
- Informatik 2 – Lehrwerk für die Oberstufe (erhältlich ab 2/2012 bei Schöningh)
- Materialien zu den zentralen Abiturprüfungen ab 2012 (bei Standardsicherung NRW)

¹ Der UML2-Standard sieht bei Assoziationen die Angabe der Multiplizität auf beiden Seiten vor. Da dies für das Zentralabitur nicht notwendig wird, kann die Angabe entfallen.