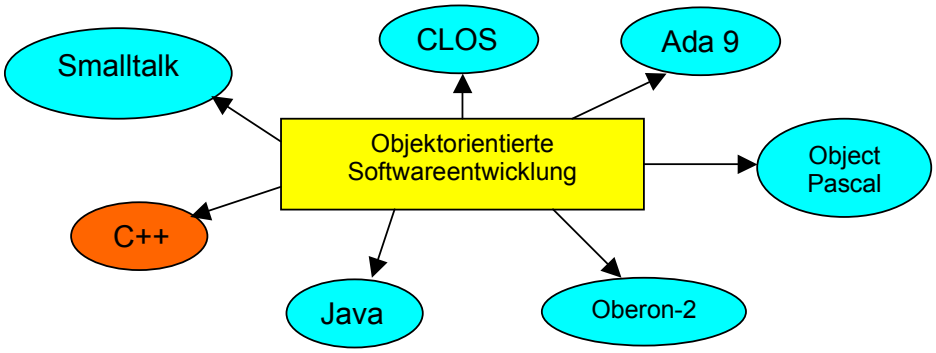
Objektorientierte Softwareentwicklung

Objektorientierte Softwareentwicklung



Frage: Die Bibliothek der Fachhochschule möchte eine Software entwickeln lassen, die es erlaubt Bestellungen, Ausleihe/Rückgabe und Mahnungen elektronisch zu verarbeiten. Wie gehen Sie vor?

Ihre Antwort:

Erkenntnis:

- komplexe Angelegenheit, mit „divide et impera“ zu lösen (VHIT =Vom Hirn Ins Terminal funktioniert nicht)
- müssen abstrahieren
- Projektplanung mit Phasen notwendig

Abstraktion

Definition: Abstraktion = „auf zufällige Einzelheiten verzichten, begrifflich zusammengefasste Darstellung“ (Duden)

Abstraktion ist eine vereinfachte Beschreibung, welche

1. sich auf wichtige Eigenschaften konzentriert
2. unwichtige Eigenschaften weglässt

Eine Abstraktion ist abhängig vom Standpunkt des Betrachters (bzw. Aufgabe der Applikation).

Beispiel: Auto (zu abstrahierendes Objekt)

1. Abstraktion Spielzeugauto
   Obwohl das Spielzeugauto (Modellauto) nicht mal einen Motor aufweist, ist es für Kinder die damit spielen ein vollwertiges Auto, da das Spielzeugauto alle Eigenschaften aufweist, die für sie wichtig sind: man kann damit fahren, es parkieren, Unfälle bauen, es verkaufen usw.
2. Auto aus Sicht eines Autohändlers
   Wichtige Eigenschaften sind Marke, Modell, Farbe, Jahrgang, Neuwagen/Occasion, Marge, Neupreis, letztes Vorführdatum, Marktpreis
3. Auto aus Sicht des Automechanikers
   Wichtig ist letzter Ölwechsel, auszuführende Reparaturarbeiten

Phasen

=	=	=
WAS	WIE	REALIESIERUNG

Beispiel: Hausbau

1. **Analyse (WAS):** Der Architekt versucht herauszufinden, was wir für ein Haus wollen, wie viele Zimmer, welche Art von Zimmer, sonstige Räume wie Stube, Bad, Schwimmbad welchen Stil uns gefällt usw.
2. **Design (WIE):** In dieser Phase werden konkrete Baupläne gezeichnet. Es wird festgelegt welche Materialen wir benötigen, wie die Grundmauer aussieht usw.
3. **Implementation (REALISIERUNG):** Der Bauleute und der Bagger treffen ein und es entsteht das konkrete Haus gemäss Bauplan

Analyse

- versucht Teil der realen Welt in einem Modell darzustellen (Abläufe, Anforderungen, Zusammenhänge usw.)
- hilft eine systematische Übersicht der realen Welt zu geben

Vorsicht: Analyse beschäftigt sich nicht mit WIE das Modell realisiert werden kann! Das ist Sache der Designphase.

UML (Unified Modeling Language)

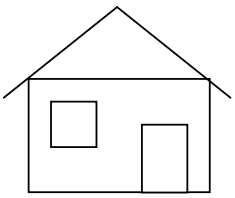
Ein Modell für die Abbildung der realen Welt ist die Uniform Modeling Language, abgekürzt UML.

- ist ein Standard
- ist eine Methode, welche allgemein benützt werden kann. Im Vordergrund stand jedoch die Informatik
- ist ein mächtiges Hilfsmittel für Analyse und spätere Phasen

Objekt

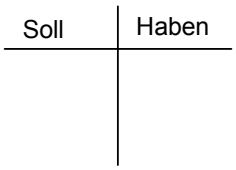
Ein Objekt ist eine Abstraktion von etwas Realem oder Erfundenem:

Physische Entität



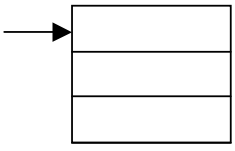
Objekt
Haus

Konzeptionelle Entität



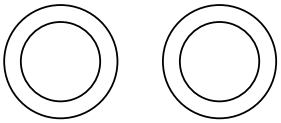
Objekt
Konto

Software Konstrukt



Objekt
Stack

Ereignis



Objekt
Eheschliessung

Eigenschaften von Objekten

Jedes Objekt hat die folgenden drei wichtigen Eigenschaften:

Status (Attribut):	aktueller Zustand aller Attribute und Beziehungen
Verhalten (Methoden):	bestimmt, was das Objekt „machen“ kann und was mit ihm gemacht werden kann.
Identität:	Eigenschaft die ein Objekt von anderen Objekten unterscheidbar macht, auch wenn diese Objekte zur gleichen Klasse gehören und gleichen Status haben.

Beispiel: Konto

KontoMannhart

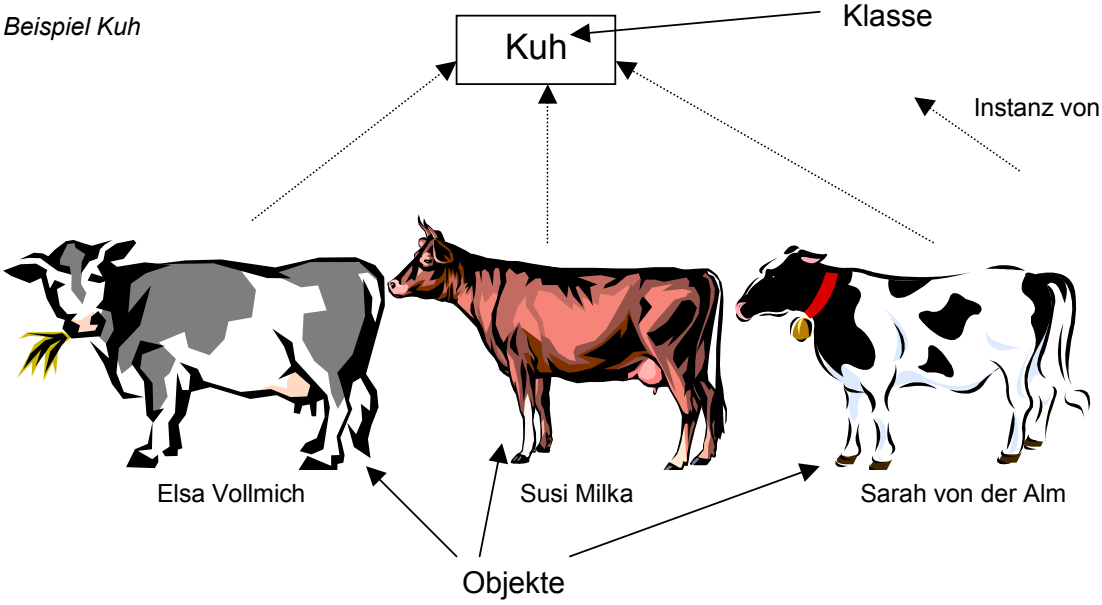
Anfangsbestand = 1000
aktueller Stand = 300

Methoden:
geldAbheben(betrag)
kontoDrucken()

Klassen

Aus ihren Programmierkenntnissen von C++ wissen Sie, das eine Variable verschiedene Werte besitzen kann. Zu jeder Variablen gehört ein Datentyp. Beispiel: `int i; i = 5; i` ist die Variable mit dem Wert 5, der dazugehörende Typ ist `int`.
Ähnlich verhält es sich mit Objekten. Objekte können verschiedene Zustände besitzen und gehören zu einem Datentyp. In der Objektorientierten Sprache wird nicht mehr von Datentyp oder Typ gesprochen, sondern von Klasse. Beispiel: `IntStack st1, st2; IntStack` ist der *Klassenname* (Typ) und `st1, st2` sind *Objekte* (Variablen).

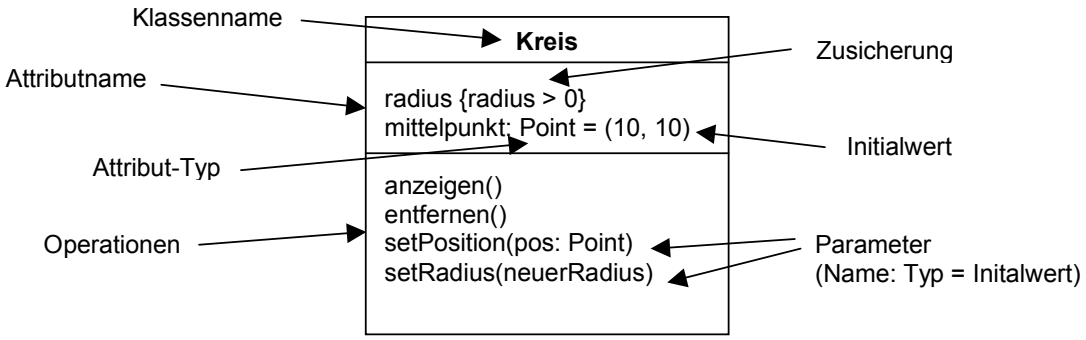
Beispiel Kuh



Definition von Klasse: Eine Klasse beschreibt die Struktur und das Verhalten einer Menge gleichartiger Objekte.

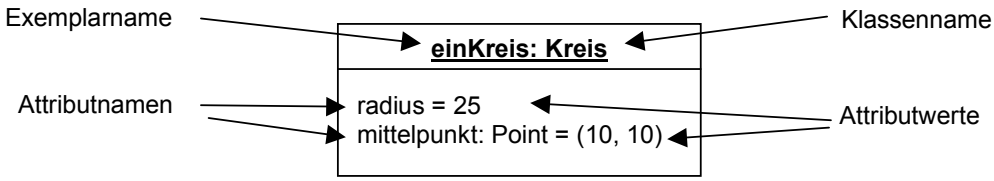
Definition von Objekt: Ein Objekt ist eine zur Ausführungszeit vorhandene und für ihre Attribute Speicher allozierte Variable, die sich entsprechend dem Protokoll ihrer Klasse verhält.

UML Notation Klasse

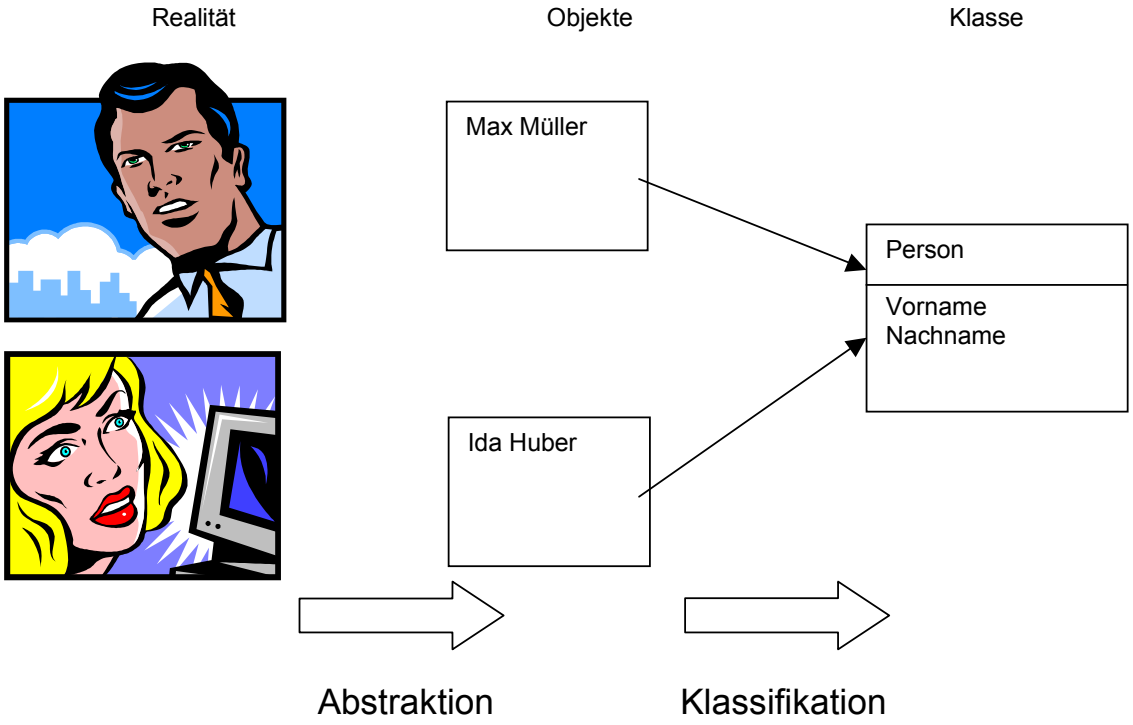


Weggelassen werden implizite Methode wie setzen und lesen von Attributen usw.

UML Notation Objekt



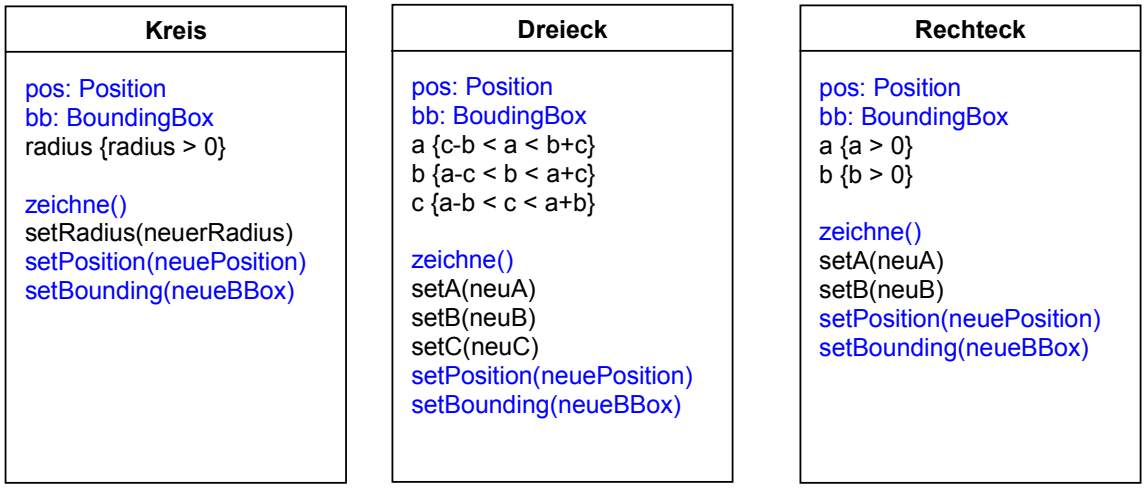
Abstraktion und Klassifikation



Wichtig: Abstraktion und Klassifikation wird meistens gleichzeitig gemacht.

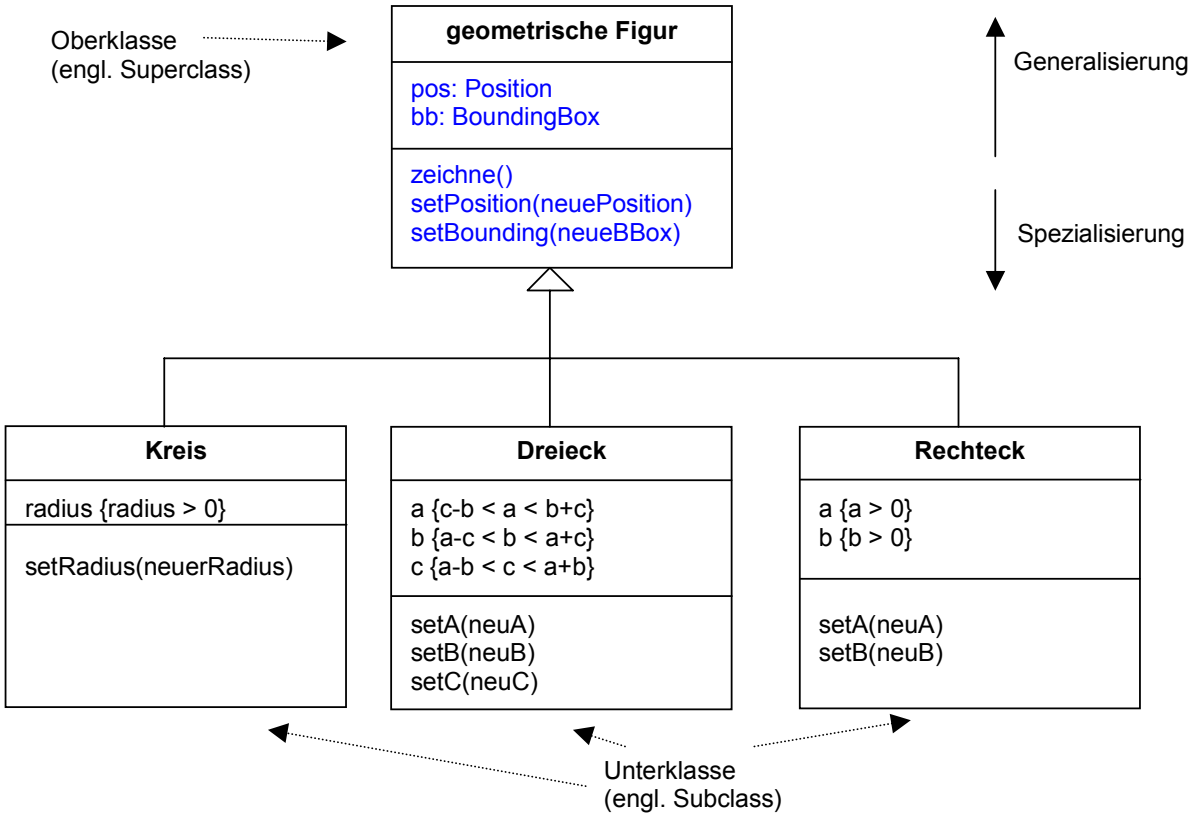
IsA Beziehung

Wir möchten eine Software schreiben, mit der wir auf dem Bildschirm 2 dimensionale geometrische Figuren zeichnen können. Wir unterstützen in der ersten Version folgende Figuren: Kreis, Dreieck und Rechteck. Die dazugehörenden Klassen werden in UML wie folgt dargestellt:



In den drei Klassen kommen die Attribute `pos` (Position), `bb` (Bounding Box) und die Methoden `zeichne(), setBounding(neueBBox)` und `setPosition(neuePosition)` vor. Es stellt sich die Frage, ob es nicht Sinn macht, diese in einer Klasse von `Geometrische Figur` zusammenzufassen und eine Beziehung zu `Kreis`, `Dreieck` und `Rechteck` herzustellen. Ein Kreis ist eine geometrische Figur. Ein Dreieck und Rechteck sind ebenfalls geometrische Figuren. Von der Klasse `geometrische Figur` betrachtet, sind `Kreis`, `Rechteck` und `Dreieck` eine Spezialisierung von `geometrische Figur`. Von der Klasse `Kreis`, `Dreieck` und `Rechteck` betrachtet ist die Klasse `geometrisches Figur` eine *Generalisierung*. In der objektorientierten Welt wird die Klasse `geometrische Figur` *Oberklasse* (engl. Superclass) von der Klasse `Kreis`, `Dreieck` und `Rechteck` genannt. Die Klassen `Kreis`, `Dreieck` und `Rechteck` heissen *Unterklasse* (engl. Subclass) von der Klasse `geometrische Figur`.

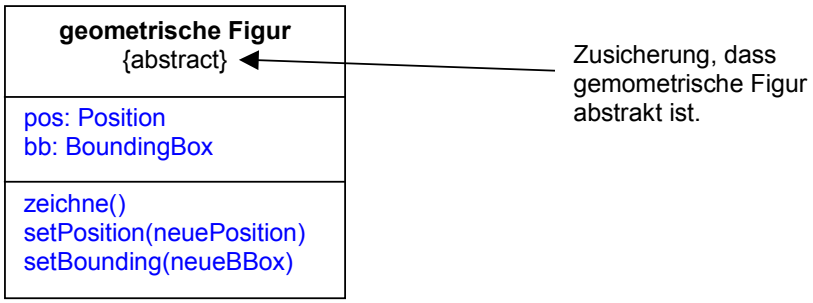
Eine „ist“ Beziehung wird *„IsA“* Beziehung (Generalisierung – Spezialisierung) genannt. Sämtliche Unterklassen weisen alle Eigenschaften der Oberklasse auf und eventuell kommen neue dazu. Dies wird auch *Vererbung* genannt.



Abstrakte Klassen

Im obigen Beispiel haben wir die Klasse `geometrische Figur` eingeführt, welche die Position `pos`, die Bounding Box `bb` als Attribute und die Methoden `zeichne()`, `setPosition(neuePosition)` und `setBounding(neueBBox)` aufweist. Was ist nun ein Objekt von dieser Klasse? Ein Punkt? Was wäre die Bounding Box des Punktes? Wir stellen 2 dimensionale Figuren dar und der Punkt ist 1 dimensional. Es macht somit keinen Sinn Objekte von der Klasse geometrisches Figur zu erzeugen. Von der Klasse `geometrische Figur` gibt es keine Objekte! Eine solche Klasse nennt man *abstrakte Klasse*.

Notation:



Assoziation

Objekte können in Beziehungen zu anderen Objekten stehen.

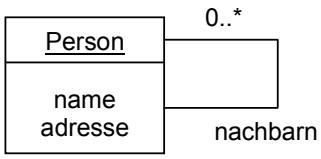
Beispiel: Fenster – geometrische Figur

Ein geometrisches Objekt wird in einem Fenster dargestellt. Ein Fenster stellt 0 oder beliebig viele geometrische Figuren dar.

Beispiel: Kunde – Artikel

Ein Kunde „bestellt“ 0 oder beliebig viele Artikel. Ein Artikel kann durch mehrere Kunden „bestellt“ werden. „bestellt“ ist der Name der Assoziation.

Beispiel: Person – Nachbar



Eine Person hat (kennt) 0 oder beliebig viele Nachbarn. „nachbarn“ ist der Name der Assoziation.

In der Regel ist eine Assoziation symmetrisch. D.h. die beteiligten Klassen bzw. Objekte kennen sich grundsätzlich gegenseitig. Falls dies nicht nötig ist, spricht man von einer gerichteten Assoziation.

Definition: Eine *Assoziation* ist eine Beziehung zwischen verschiedenen Objekten einer oder mehreren Klassen.

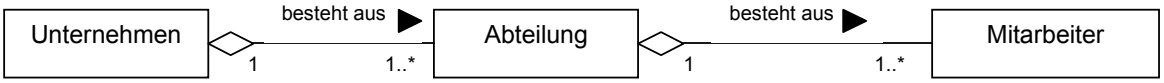
Aggregation – Has Beziehung

Eine Aggregation ist eine besondere Variante der Assoziation. Es handelt sich um eine Beziehung zwischen Klassen, jedoch mit der Besonderheit, dass die Klassen zueinander in Beziehung stehen, wie ein ganzes zu seinen Teilen.

Beispiel: Segelboot

Segelboot = Rumpf + Segel + Motor

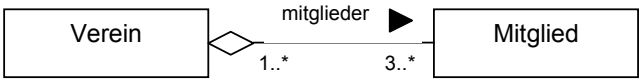
Beispiel Firma



Eine Firma besteht (hat) eine Abteilung. Die Abteilung besteht (hat) Mitarbeiter

Aus den obigen Beispielen geht hervor, dass ein Objekt aus anderen Objekten besteht, diese hat. Diese „hat“ Beziehung nennt man Has.

Beispiel Verein



Ein Verein besteht aus mindestens 3 Mitgliedern. Ein Mitglied gehört zu mindestens 1 Verein. „mitglieder“ ist der Name der Has Beziehung

Komposition

Beispiel: Rechnung

Eine Rechnung hat mindestens eine Rechnungsposition. Eine Rechnungsposition gehört zu einer Rechnung.

Nehmen wir an, wir löschen eine Rechnung. Was geschieht nun mit den Rechnungspositionen? Diese müssen auch gelöscht werden, denn eine Rechnungsposition gehört genau zu einer Rechnung. Da diese nicht mehr existiert muss diese ebenfalls gelöscht werden. Eine solche Beziehung wird Komposition genannt.