Modellierung

Die Objektorientierung hat nicht nur mit den Konzepten Klasse und Objekt neue strukturelle Inhalte in die Informatik gebracht, sondern auch bei der Analyse und beim Entwurf neue Konzepte entwickelt.

Entwurfsmuster

Einerseits hat man sich mit Entwurfsmuster beschäftigt. Dazu gibt es ein Standardwerk von Gamma[1] u.a. , mit einigen dieser Entwurfsmuster werden wir uns noch beschäftigen.

UML

Wie schon vorher erwähnt, hat sich in der OO für die Kennzeichnung von Klassenbeziehungen inzwischen ein Quasistandard entwickelt (und wird noch weiter entwickelt), der mit ***UML*** = ***U***nified ***M***odeling ***L***anguage bezeichnet wird. Dazu gibt es ein Standardwerk von Bernd Oesterreich[2].

Beziehungstypen

U.a. beschäftigt sich die UML mit Beziehungstypen. Einige von ihnen untersuchen wir auch in diesem Kurs. Wir haben – orientiert an den Möglichkeiten, die BlueJ zur Zeit bietet – uns bisher nur mit den beiden Beziehungstypen erbt – Beziehung (auch als „ist ein -“ bezeichnet) und Nutzerbeziehung (auch als „hat ein -“ bezeichnet) beschäftigt. Das wollen wir nun etwas genauer untersuchen.

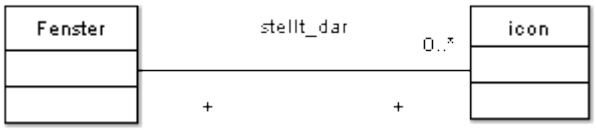
Assoziation und Aggregation

Assoziation

Die beiden Begriffe beschreiben Beziehungstypen. Es geht dabei um Beziehungen zwischen Objekten. Die Objekte werden in vielen Anwendungsfällen nicht derselben Klasse angehören, aber sie können derselben Klasse angehören.
Untersucht man z.B. die Oberflächen, die Windows oder KDE o.ä. uns anbieten, dann weisen die *icons* diesen Beziehungstyp auf: Die Fensteroberfläche stellt die icons dar. „*stellt dar*“ ist dabei eine sprachliche Beschreibung der hier vorliegenden Beziehung, genauso, wie wir sie oben bei „*ist ein*“ und „*hat ein*“ kennengelernt haben. Die Assoziation beschreibt davon die Nutzerbeziehungen.

Fensteroberfläche hat icons

Betrachten wir wieder die Fensteroberfläche, dann können wir feststellen, dass wir über die Oberfläche sagen können sie „stellt dar“ – wie oft eben nicht einen, sondern mehrere – icons.



Ein Fahrrad „hat einen“ Fahrer, eine Abteilung einer Firma „hat einen“ Mitarbeiter usw. Andererseits hat ein Fahrrad nicht nur einen Fahrer, es *besteht aus* sehr vielen Einzelteilen. Damit kommen wir auf einen Spezialfall einer Assoziation, die Aggregation.

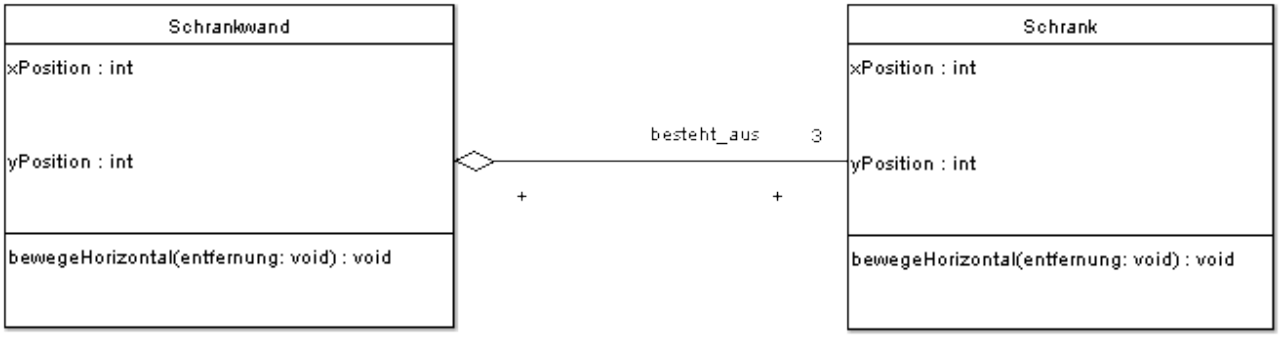
1 Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides: Entwurfsmuster – Elemente wiederverwendbarer Software ; Addison – Wesley ; ISBN 3-89319-950-0
2 Bernd Oesterreich: Objektorientierte Softwareentwicklung – Analyse und Design mit der UML ; Oldenbourg ; ISBN 3-486-24787-5

Aggregation

Die Aggregation ist ein besonders interessanter Spezialfall, Bernd Oesterreich bezeichnet sie als Teil – Ganzes – Beziehung oder „hat“ - Beziehung. Oesterreich schreibt in seinem Buch:
Eine Aggregation ist die Zusammensetzung eines Objektes aus einer Menge von Einzelteilen. Ein Auto ist beispielsweise eine Aggregation von Rädern, Motor, Lenkrad usw. Auch diese Teile sind ggf. wieder Aggregationen: eine Bremse besteht aus ...
Statt von einzelnen Aggregationen ist manchmal von Teile-Ganzes-Hierarchien die Rede.

Die grafische Kennzeichnung der Beziehung im Klassendiagramm erfolgt mit einer einfachen Linie und einer offenen Raute am Ganzes – Ende. Dabei kann man an den Enden die möglichen Anzahlen der Beteiligten darstellen.

Schrankwand



(Diagramm erstellt mit ArgoUML)

Eine Schrankwand ist ein Beispielfall für diesen speziellen Beziehungstyp Aggregation. Die Schrankwand „*hat ein*“ – hier wieder in der Regel mehrere – Schrankelemente. Das klingt etwas holprig. Wir würden nämlich eher sagen: Eine Schrankwand „*besteht aus*“ z.B. drei Schrankelementen. Ist das Ende mit einer einzelnen Zahl gekennzeichnet, wie hier mit der 3, dann müssen es immer 3 Schränke sein, aus denen eine Schrankwand besteht. Sind es mindestens 1 und können theoretisch beliebig viele sein, dann gibt man an dem Ende 1..* an, bei höchstens drei würde man 0..3 angeben.

Komposition

Für die Verwendung des Begriffes Aggregation ist es wichtig zu klären, ob die benutzten Klassen auch unabhängig von der benutzenden Klasse existenzfähig sind. Falls diese Unabhängigkeit nicht gegeben ist, spricht man speziell von einer *Komposition*.
Ob die einzelnen Schrankelemente einer Schrankwand für sich „lebensfähig“ sind, kann man durchaus bezweifeln. Daher wäre die Beziehung zwischen beiden ggf. eine Komposition und man müsste statt der offenen Raute eine geschlossene verwenden.

Komposition und Kompositum

Eine unglückliche Begriffsähnlichkeit tritt bei den beiden Begriffen Komposition und dem Entwurfsmuster Kompositum auf. Es handelt sich nicht nur bei einer Komposition um eine besondere Form von Aggregation. Auch bei einem Kompositum geht es um eine spezielle Form von Aggregation:

Oesterreich[3]: *Mit dem Kompositum – Muster werden baumartige Aggregationen hergestellt (zusammengesetzt), die sowohl als einzelne Objekt als auch als Zusammmensetzung von Objekten in gleicher Weise benutzt werden können*.

Im Gegensatz zur Komposition können die Elemente eines Kompositum durchaus selbstständig lebensfähig sein. Das besondere Merkmal eines Kompositum ist, dass auch die ganze Aggregation (im Prinzip) dieselbe Schnittstelle aufweist, wie ihre Elemente. Das gilt auch rekursiv geschachtelt. Angewandt z.B. auf eine Schrankwand bedeutet das, dass wir sie als Ganzes verschieben, drehen oder umfärben können, selbst Komposita geschachtelt in andere Komposita werden dann mit verschoben usw.
Wegen der möglichen Selbstbezüglichkeit ist die grafische Darstellung des Kompositummusters etwas komplizierter, u.a. enthält sie auch „ist ein“ – Beziehungen.

Anwendung des Kompositummusters ?

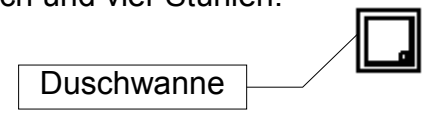
Weder unser erster Entwurf für die Schrankwand noch der zweite sind wie das Entwurfsmuster Kompositum realisiert. Die Behandlung von Entwurfsmustern allgemein und des Entwurfsmusters Kompositum speziell steht noch aus!

Anwendungsfälle von Aggregationen
Zunächst einmal wollen wir uns mit einfachen Anwendungsfällen von Aggregationen beschäftigen, ohne über die Entwurfsmuster nachzudenken. Dabei wollen wir auch nach Fällen von Kompositionen sehen.

Aufgabe:

Untersuchen Sie die folgenden Beispiele daraufhin, ob sie Assoziationen, Aggregationen oder Kompositionen enthalten:

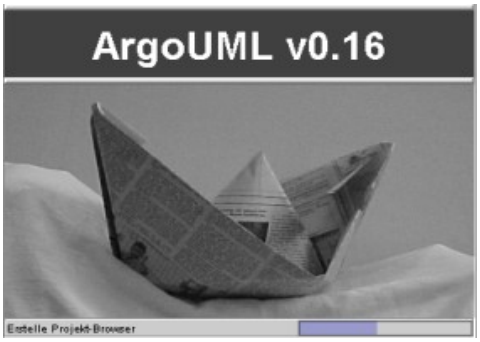
- Eine Sitzgruppe, bestehend aus einem quadratischen Tisch und vier Stühlen.
- Eine Duschwanne, bestehend aus drei Quadraten
- Ein Kochherd, der einen Benutzer benötigt.
- Ein Kochherd, der eine Küche benötigt.
- Auto, Fahrrad, Fahrzeug, PKW, LKW, Reifen, Türen, Fahrer



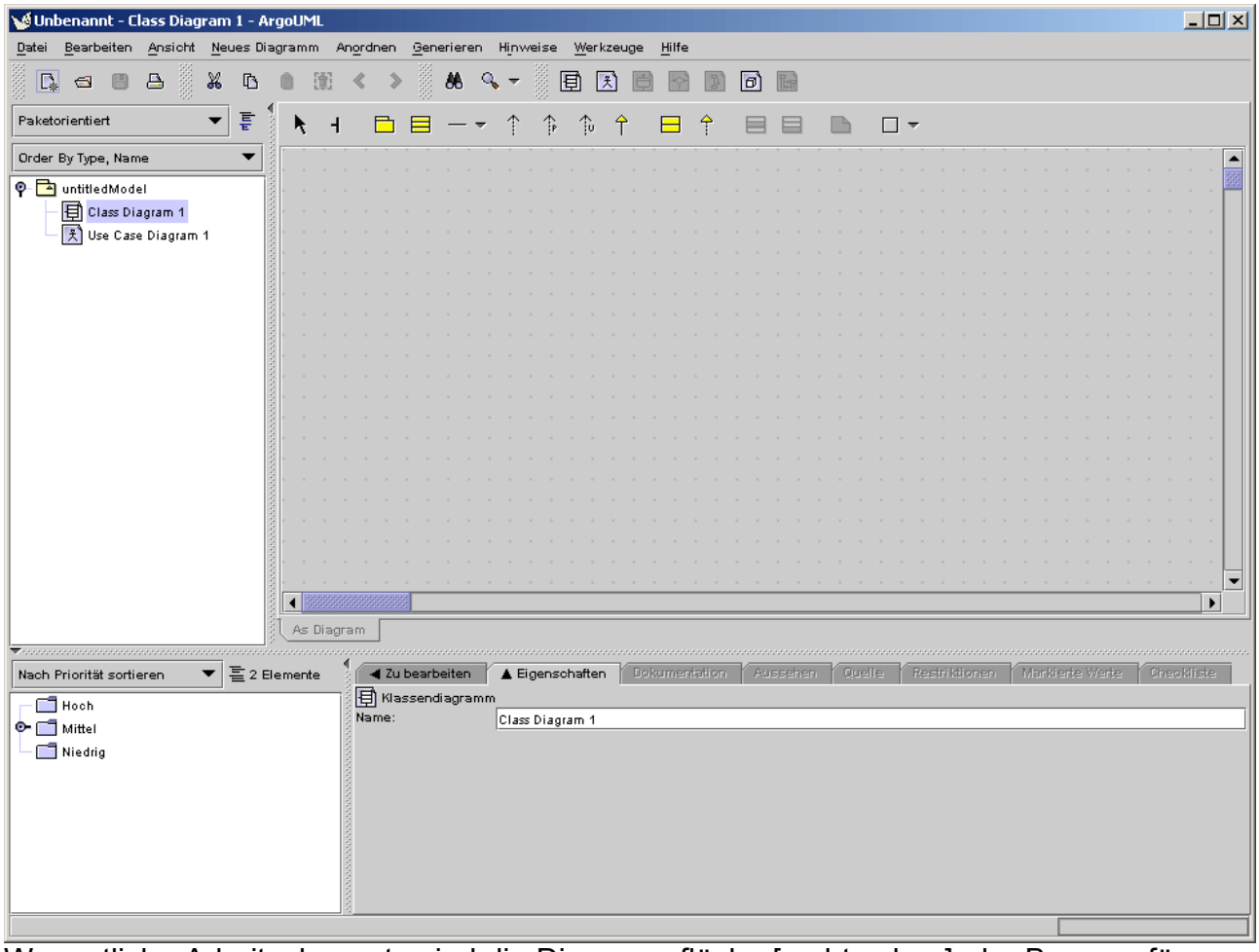
Zeichnen Sie die zugehörigen UML – Diagramme (auch in BlueJ – Variante) !

3Siehe dazu auch Gamma: „Entwurfsmuster“ ab S.239.

Ein Werkzeug für UML



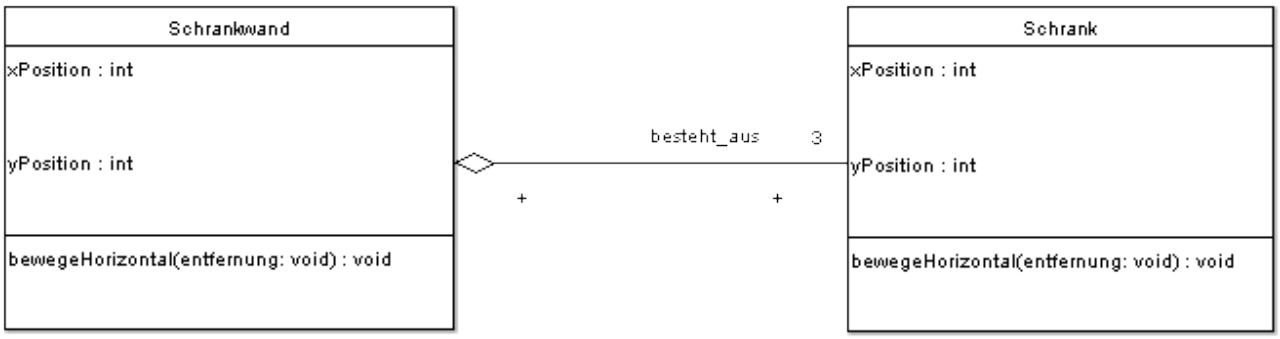
Sucht man nach Werkzeugen für das Erstellen von UML – Diagrammen, dann findet man zur Zeit viele neue Beispiele. Vielfach diskutiert wird auch die Frage, ob die UML inzwischen oder künftig auch den Ansatz einer Programmiersprache darstellt, d.h. „man programmiert in UML“. Ich will auf diese Frage hier nicht eingehen, das auch, weil die Diskussion mit ähnlichen Glaubens-ansätzen geführt wird wie die früheren, z.B. die, ob „man in BASIC programmieren darf“ oder ob es PASCAL sein muss. Deshalb stellt sich die Frage, ob ein solches Werkzeug für uns im Unterricht eine einfache Darstellung ermöglicht und ggf. leicht zu einem umsetzbaren Projekt führt.
Die folgenden Bemerkungen beziehen sich auf das Werkzeug ArgoUML, das kostenlos im Netz bereitgestellt [http://argouml.tigris.org] wird, selbst in JAVA programmiert ist, also nicht nur in Windows – Umgebungen als .jar- file gestartet werden kann. Die Oberfläche sieht nach dem Start so aus:



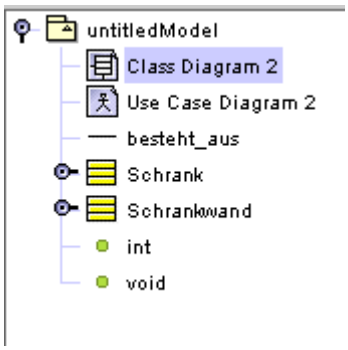
Wesentliche Arbeitselemente sind die Diagrammfläche [rechts oben], der Browser für Diagrammobjekte [links oben], das Sortierungsfenster [links unten] und das Definitionsfenster [rechts unten].

Beispiel Schrankwand

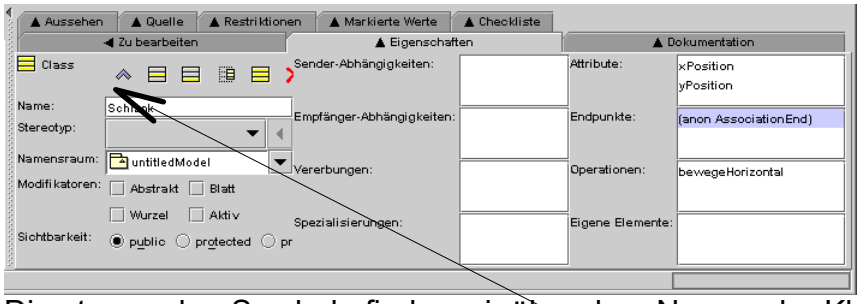
Das Beispiel Schrankwand mit nur wenigen Attributen und Methoden sieht dann z.B. so aus:



Das zugehörige Fenster für die Modellobjekte zeigt die enthaltenen Klassen Schrank und Schrankwand, die Beziehung „besteht_aus“ und die auftretenden Datentypen void und int .

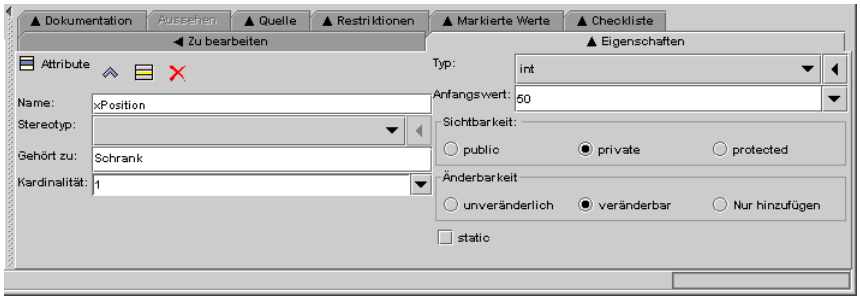


Im Definitionsfenster finden wir bei markiertem Klassensymbol die eigenschaften und Methoden der Klasse. Hier können wir sie auch bearbeiten.



Die steuernden Symbole finden wir über dem Namen der Klasse. Der kleine Winkel dient zum Wechsel nach oben in der Bearbeitungshierarchie, das nachfolgende Symbol ist für das Einfügen eines Attributes, dann das Symbol für Methoden, das Symbol für das Einfügen innerer Klassen, dann das Symbol für neue Klassen und das Symbol für das Löschen.

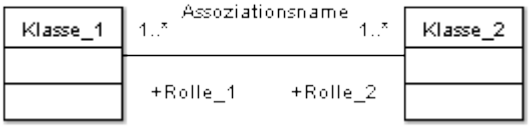
Bearbeitet man ein Attribut, wie z.B. das Attribut xPosition, dann sieht dies Fenster etwas ander aus. Wir bekommen die Möglichkeit, den Typ, einen Startwert und die Sichtbarkeit zu definieren.



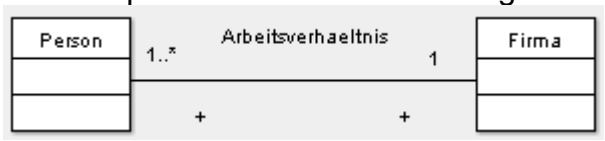
Hat man das Modell vervollständigt, kann man es in Programmcode konvertieren. Dazu geht man im Menüpunkt Generieren auf Alle Klassen Generieren, wählt die verfügbaren Klassen und das Ausgabeverzeichnis. Viel weiter bin ich noch nicht gekommen. Weitere Hilfen wären interessant.

Einige genauere Betrachtungen zu Assoziationen

In UML 2 glasklar (Rupp, Hahn, Queins, Jeckle, Zengler; Verlag:Hanser) finden wir neben allgemeinen Beschreibungen der Darstellung von Assoziationen in Klassendiagrammen wie dem folgenden



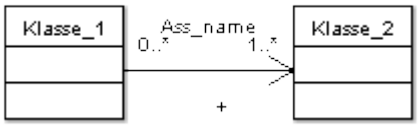
auch einige Analysen von Beispielfällen. Beim folgenden Beispiel wird untersucht, welche Rolle die erzeugten Objekte entsprechend dem Klassendiagramm[4] erhalten.



Zur Laufzeit des Programms werden nun Exemplare erzeugt, aber welche?
Das Diagramm sagt aus, dass eine Assoziation `Arbeitsverhaeltnis` voraussetzt, dass ein „Exemplar“ Person mit einem „Exemplar“ Firma verbunden ist. „1..*“ bedeutet, dass mehrere Person – Objekte ein Arbeitsverhaeltnis zu derselben Firma haben können, aber nicht zu verschiedenen Firmen. (Ob das eine realistische Modellierung ist, soll hier nicht untersucht werden.)
Gibt es drei Arbeitnehmer der Firma „Producki“ mit den Namen „Hans“, „Janina“ und „Ulf“, dann müssen, wenn die Assoziation angelegt wird, vier Objekte erzeugt werden, eben

- ein Objekt der Klasse Firma, bei dem ein Attribut den Wert „Producki“ hat,
- ein Objekt der Klasse Person, bei dem ein Attribut den Wert „Hans“hat,
- ein Objekt der Klasse Person, bei dem ein Attribut den Wert „Janina“ hat und
- ein Objekt der Klasse Person, bei dem ein Attribut den Wert „Ulf“ hat.

Die Verbindungen werden als Links (wie in html) bezeichnet. Eine Assoziation beschreibt im Klassendiagramm die auftretenden Links. In dem Beispiel wird also irgendein Exemplar von Person mit genau (s.o.) einem Exemplar von Firma durch einen Link verbunden.
Rollen beschreiben, welche Rolle das eine Objekt für das andere spielt. Im Beispiel ist die Person Arbeitnehmer der Firma, die wiederum Arbeitgeber, so dass die Enden entsprechend gekennzeichnet sein können.
Ich möchte ausdrücklich darauf hinweisen, dass für unsere Modellierung sparsames Kennzeichnen sinnvoll sein kann. Je sparsamer ausgefüllt desto lesbarer werden die Diagramme! Man nimmt dann in Kauf, dass ein Teil der Aussagekraft verloren geht.
Genauere Informationen entnehme man dem o.a. Buch. Mir scheint allein eine Kennzeichnung noch wichtig, mit der ausgedrückt werden kann, dass ein Objekt das Objekt, mit dem es mit dem Link verbunden ist, kennt. Dafür benutzt man einen Pfeil:



Jede Instanz von Klasse_1 kennt seine Instanz von Klasse_2.

4 Das Klassendiagramm wurde mit ArgoUML erstellt und stellt die 1 am rechten Ende der Beziehung nicht dar. Ich habe sie hier mit einem Grafikprogramm eingefügt.