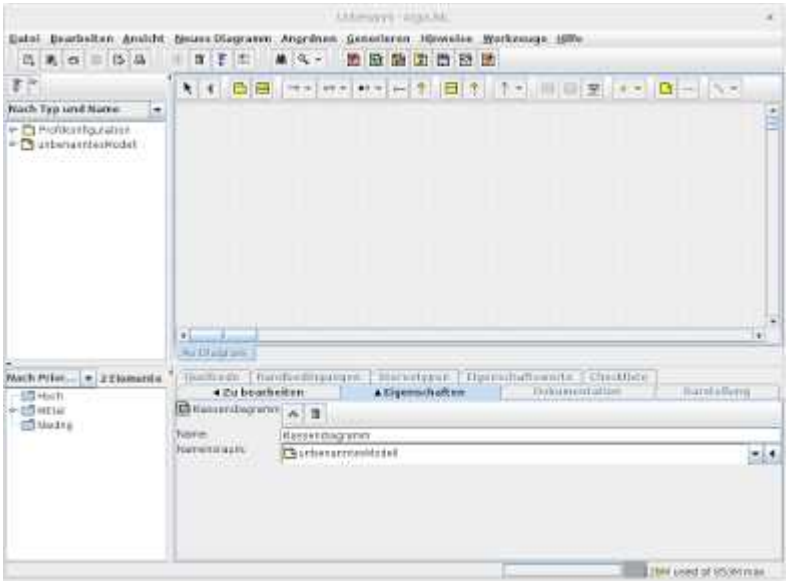
Klassendiagramme und Modellierung[1]

Im Unterschied zur Entwicklung von OO Projekten mit BlueJ arbeiten wir bei der Entwicklung mit IDLE für Python-Projekte rein textbasiert. Das Besondere an der BlueJ-Umgebung ist, dass direkt aus dem Klassendiagramm (classbrowser) ein rudimentärer Programmtext erzeugt werden kann. In der OO hat sich für die Kennzeichnung von Klassenbeziehungen[2] inzwischen ein Quasistandard entwickelt (und wird noch weiter entwickelt), der mit ***UML*** = ***U***nified ***M***odeling ***L***anguage bezeichnet wird. Leider folgt BlueJ bei der Kennzeichnung nicht diesem UML-Standard.

ArgoUML



Bei der Suche nach entsprechenden Möglichkeiten für Python[3] habe ich zunächst bei dem Tool ArgoUML [http://argouml.tigris.org/] nachgesehen, das ich bereits unter Java eingesetzt habe.

Leider haben die Entwickler zwar einmal angekündigt, sie würden das Generieren von Python-Programmcode einbauen, haben das aber bisher nicht umgesetzt.

Es kann allerdings eine interessante Aufgabe sein, zu einem mit ArgoUML erstellten Klassendiagramm den Java-Code generieren zu lassen und diesen dann in den entsprechenden Python-Code zu ändern. Dabei werden die konzeptionellen Gemeinsamkeiten, aber auch die Unterschiede der Syntax der beiden Sprachen deutlich, von denen einige wichtige in der Tabelle im Anhang angegeben sind.

Trotz dieser Unterschiede sind die Texte nicht so verschieden, dass sich dieser Weg gar nicht lohnt. [siehe DIA-Beispiel unten]

1 In diesem Text wird nicht auf das von mir selbst mit Python und wxPython entwickelte Tool eingegangen. Download: http://claus.albowski.de unter Software zu finden. Wir setzen es im Kurs ein, nicht die hier vorgestellten Tools.

2 UML definiert nicht nur die Darstellung von Klassenbeziehungen, sondern auch andere Modellierungsaspekte wie z.B. Sequenzen; hier geht es zunächst jedoch nur um Klassenbeziehungen.

3 Siehe z.B. http://wiki.python.de/PythonUml
Dort heißt es u.a.:
"Im großen und ganzen gibt es zwei verschiedene Arten von Programmen: Es gibt "reine" Malwerkzeuge wie DIA oder Visio, die UML unter anderem anbieten, und es gibt Tools die sich völlig auf UML spezialisiert haben. Die Python- Unterstützung ist jedoch nur selten zu finden. Zwar können Umbrello, Object Domain und Enterprise Architect wohl Python-Code erzeugen, ausprobiert habe ich es jedoch nur mit Umbrello. Für ArgoUML ist eine Python-Unterstützung geplant, die Entwicklung ist aber wohl eingeschlafen. Das Programm ist aber IMHO auf jeden Fall einen Blick wert, da es auf Unstimmigkeiten und Fehler in den Diagrammen hinweist und (deutschsprachige) Lösungsvorschläge anbietet. Zudem ist es kostenfrei. Ein echtes Roundtrip engineering, wie es viele Produkte für Java anbieten, ist AFAIK noch mit keinem für Python erhältlichen Produkt möglich. In den meisten Fällen lässt sich aber XMI erzeugen."
Zu XMI siehe z.B. http://www.jeckle.de/xmi.htm. Wir sollten aber "kein neues Fass aufmachen".

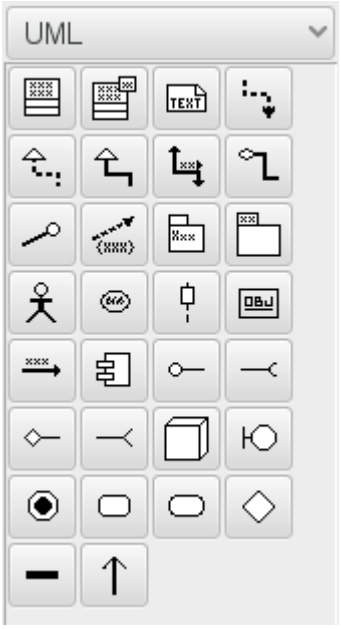
DIA

Mit DIA steht unter allen Betriebssystemen ein allgemeines Grafik-Werkzeug zum Erstellen von Diagrammen zur Verfügung, das auch UML-Diagramme erzeugt. DIA bietet [leider mit einigen unkomfortablen Vorbereitungen bei Windowsinstallationen[1] von DIA] mit ***PyDia*** auch das direkte Generieren von [nicht nur] Python-Code an. Ein Beispiel zum von DIA generierten Code: Die Methode BewegeHorizontal wird als Java-Code generiert:

```
/**
 * @param weite
 * @return void
 */
public void BewegeHorizontal(Double weite) {
      // TODO Auto-generated method stub
}
```

Als Python-Code:

```
def BewegeHorizontal (self, weite) :
      # returns
      pass
```



DIA-Elemente für UML

Bei dieser Darstellung hat man in beiden Fällen also die Signatur richtig bekommen. Den eigentlichen Programmtext zu schreiben, sollte danach leicht fallen.
Eine einfache Variante bietet unter Windows das Programm ***dia2code*** [http://dia-installer.de/dia2code/index.html]. Die generierten Python-Programme zeigen einige Unterschiede. Auffallend ist, dass bei der PyDia-Variante der Kommentar als Dokumentationsstring eingetragen wird, andererseits dia2code die Sichtbarkeit richtig verarbeitet, so dass das Attribut x als __x eingetragen wird. Allerdings wird es nicht als Attribut realisiert, sondern als Klassenvariable und der Konstruktor fehlt. Daher ist eher die PyDia-Variante zu empfehlen.

Klassendiagramme

Erstellt man mit diesen Werkzeugen Klassendiagramme, dann lassen sich den Klassen in der Regel Eigenschaften wie beispielsweise *abstract* zuweisen[2]. Weiterhin kann man Attribute mit ihren Typen[3] und ihrer Sichtbarkeit angeben. Auch die Methoden [Operationen] können angegeben werden, dabei werden zu den Parametern ähnlich der Situation bei den Attributen ihre Typen erwartet wie zum Rückgabewert. Attribute werden in der zweiten Zeile des Klassensymbols dargestellt, Methoden in der dritten.

1 DIA braucht unter Windows die passende alte Pythonversion 2.3, die man vor der Installation der neuesten DIA-Version installieren muss. Dabei sollte man darauf achten, dass man bei den Einstellungen [unter "weitere Einstellungen"] das Registrieren dieser Version als Standard für Python-Anwendungen wegklickt. Dann lassen sich Dia und die neueste Python-Version problemlos nebeneinander benutzen. Als weitere Programme müssen nach dem Installieren von DIA noch pycairo und pygtk installiert werden. Bei mir hat es mit den Versionen Python-2.3.5, pycairo-1.0.2-1.win32-py2.3 und pygtk-2.8.6-1.win32-py2.3 funktioniert.
Wichtig ist die Reihenfolge: Python 2.3 → Dia → pycairo → pygtk

2 BlueJ bietet diese Zuweisungen im Modellfenster nur in ganz kleinem Umfang. Außerdem entspricht die Darstellung leider nicht der Standarddarstellung der UML.

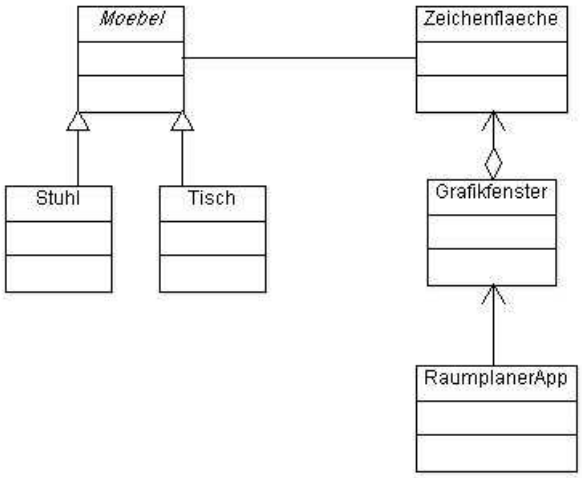
3 Typangabe und Sichtbarkeit sind prinzipiell natürlich nicht zwingend; siehe bei Python: Es gibt keine strenge Typbindung, insbesondere keine Deklaration des Typs. Es gibt kein Sichtbarkeitsprädikat.

Unser erster Modellierungsschritt mit der Klasse Moebel

Fügen wir die Klasse Moebel ein, dann sollte sie als *abstract* gekennzeichnet sein, was bei ArgoUML mit der kursiven Schreibweise geschieht[1].

Das ist aber nicht die einzige Form der Veränderung, die nach dem Einfügen der neuen Klasse Moebel notwendig ist. Zusätzlich werden Pfeile von den konkreten Möbelklassen in das Diagramm eingetragen.

Sie haben eine vorgegebene Gestaltung: Sie sind mit durchgezogener Linie und einer Pfeilspitze gezeichnet, die aus einem geschlossenen Dreieck besteht. Damit kennzeichnet das Diagramm den Beziehungstyp Vererbung [Generalisierung].



UML-Diagramm mit ArgoUML

Diese Beziehungstypen verstehen und bewerten zu können, ist ein wichtiges Ziel in diesem Bereich der Modellierung.

> Eine durchgezogene Linie mit einer geschlossenen Pfeilspitze kennzeichnet eine erbt – Beziehung.

Es gibt nicht nur Vererbung

Das vollständige oben dargestellte Diagramm enthält außerdem eine durchgezogene Linie mit einer offenen Pfeilspitze[2] [link[3]]. Diese Kennzeichnung gehört zu einem anderen Beziehungstyp, von dem es mehrere Varianten gibt. Die UML kennt also weitere Unterscheidungen von Beziehungstypen, die z.T. auch im o.a. Diagramm zu finden sind und die wir noch genauer kennen lernen werden.

> Eine durchgezogene Linie ohne oder mit einer offenen Pfeilspitze kennzeichnet eine **Assoziation** [**benutzt** – Beziehung].

Allgemein gilt für diese Beziehungen, dass es sich eigentlich nicht um Beziehungen zwischen Klassen[4] handelt wie bei der Vererbungsbeziehung, sondern um Beziehungen zwischen Objekten. Dabei benutzt ein Objekt ein anderes[5], wobei diese Beziehung in einigen Fällen beidseitig sein kann.

Den einfachen Typ dieser Assoziationsbeziehung kann man sprachlich neben "benutzt" auch "**hat**" oder "**kennt**" o.ä. beschreiben. Ein wichtiger Spezialfall dieser Beziehung lässt

1 Alternativ erfolgt die Kennzeichnung durch {abstract}.
2 Die Pfeilspitze gibt eine mögliche "Navigationsrichtung" an.
3 "… kann die Beziehung als Weg oder Kommunikationskanal verstanden werden." [Quelle: UML 2 glasklar]
4 Dennoch gilt, dass sie auf der Ebene der Klassen beschrieben werden können. Sie geben eine mögliche Beziehung an, die in einigen konkreten Fällen nicht wirklich hergestellt werden muss.
5 Das Diagramm zeigt mit der einfachen Linie zwischen der abstrakten Klasse Moebel und der Klasse Zeichenflaeche an, dass ein konkretes Moebel-Objekt das Zeichenflaechen-Objekt benutzt, um sich darzustellen.

sich genauer mit "**besteht aus**" beschreiben[1] und kennzeichnet damit eine "**Teil-Ganzes-Beziehung**" mit den beiden Varianten **Aggregation** und **Komposition**.

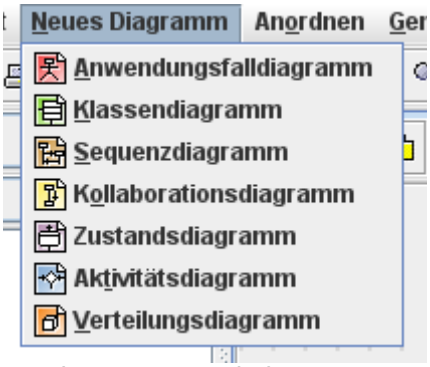
Beschränkung auf das Wesentliche

Dabei ist zu beachten, dass es sinnvoll ist, bei der Behandlung der Beziehungstypen auf die Teil-Ganzes-Beziehung einzugehen. Insbesondere in Kursen mit "Grundlegendem Niveau" kann eine Beschränkung auf das Wesentliche für das Verständnisder Schülerinnen und Schüler aber wichtiger sein als Genauigkeit in den Feinheiten.

In einem der Standardwerke der UML von Bernd Oesterreicher „Objektorientierte Softwareentwicklung – Analyse und Design mit der ...“ heißt es:
Erdrückende Vielfalt?
Die Grundkonzepte objektorientierter Softwareentwicklungsmethodik sind ausgereift und bewähren sich in der Praxis. Andererseits bietet gerade die UML einen beachtlichen Detailreichtum und ist daher durchaus mit Bedacht anzuwenden.
Die Vielfalt der Beschreibungsmöglichkeiten kann sehr erdrückend wirken, ein tieferes Verständnis für die UML-Konstrukte in allen Facetten erfordert einigen Aufwand. In einer ersten Annäherung kann man sich deshalb auf die grundlegenden Elemente beschränken. Es bleiben damit gegebenenfalls zwar semantische Lücken in der Modellierung, dennoch kann die Arbeit auf diesem Niveau in der Praxis ausreichen. Ganz abgesehen davon wird (vielerorts) überhaupt keine Systematik angewendet.

Andere Diagrammtypen

Deswegen soll hier noch einmal darauf hingewiesen werden, dass die UML viele weitere Diagrammtypen kennt. So kann beispielsweise mit Sequenzdiagrammen[2] gut der Austausch von Botschaften [messages] zwischen den beteiligten Objekten und die zeitliche und logische Reihenfolge von Abläufen modelliert werden.
Der Hamburger Rahmenplan fordert aber deren Behandlung nicht und das Raumplaner-Projekt ist so angelegt, dass es vor allem auf die im Klassendiagramm dargestellten Beziehungen ankommt.



Diagrammtypen bei ArgoUML

1 Die Beziehungstypen mit den Fachbegriffen Assoziation, Aggregation und Komposition werden später am Beispiel ausführlich betrachtet.
2 Für Sequenzdiagramme bietet DIA in der Auswahl zur UML entsprechende Objekte. Für BlueJ unter Java gibt es ein Plugin.

Anhang

Tabelle zum Vergleich Java – Python

	Java	Python
Programmblöcke	**{** -> Einrückung sollte erfolgen und dann gleich sein **}**	**xxxx(...):** **weiter...** -> Einrückung muss mit exakt gleicher Tiefe erfolgen
Typbindung	zwingend	aus dem Kontext
Deklaration	notwendig	. / .
Anweisungsabschluss	**;**	Zeilenende Bei notwendigem Umbruch Zeilenende mit \ auskommentieren
Methodenkopf	mit Sichtbarkeit und Typ	**def <Name>(self …):**
Attributzugriff	**this** muss nur bei Überdeckung verwendet werden	**self** muss bei allen Attributen verwendet werden
Konstruktoren	Überladen [overload]	genau einer mit ggf. vordefinierten Parametern