Objekte und Klassen (Teil 1)

Ausgehend von einer Anforderungsanalyse, in der die Anforderungen eines Kunden für eine zu erstellende Software gesammelt und analysiert werden, wird mittels der Objektorientierten Analyse bestimmt, welche Objekte existieren, welche Eigenschaften diese Objekte haben und wie diese Objekte miteinander in Beziehung stehen bzw. interagieren.

Dies soll im folgenden am Beispiel eines elektronischen Adress- und Telefonbuches erfolgen.

Objekte

Die Objektorientierte Analyse ermittelt Objekte sowie deren Eigenschaften und Verhaltensweisen , die sich im Rahmen der Anforderungsanalyse ergeben haben.

So könnte der Kunde beispielsweise gesagt haben:

> *„Ich möchte gerne zu meinen Geschäftspartnern deren persönliche Daten speichern. Ein Beispiel wäre mein Geschäftspartner Dr. Hugo Müller, der in 64521 Groß-Gerau in der Sudetenstraße 34 wohnt. Außerdem hat Dr. Müller noch ein Ferienhaus auf 25980 Sylt in der Norderstraße 28. Herr Dr. Müller ist unter folgenden Telefonnummern erreichbar: In der Firma unter (069) 7380 321, in Groß-Gerau unter (06151) 912345, in Sylt unter (04651) 6521 und mobil unter (0172) 3671298. Dr. Müller hat am 23.02.1967 Geburtstag. Sollten sich die Daten ändern, sollen diese natürlich angepasst werden können. Natürlich möchte ich neue Geschäftspartner hinzufügen können. Manchmal weiß ich von einem Geschäftspartner keine Adresse und/oder Telefonnummer, wenn ich diesen eintragen will.“*

Analysiert man diese Beschreibung, ergeben sich beispielsweise folgende Objekte:

1. Person-Objekt; hier: Dr. Hugo Müller, der am 23.02.1967 Geburtstag hat.
2. Adresse-Objekte; hier 64521 Groß-Gerau, Sudetenstraße 34 sowie 25980 Sylt, Norderstraße 28.
3. Telefon-Objekte: (069) 7380321, (06151) 912345 und (0172) 3671298.

Analysiert man nun die softwarerelevanten Daten, ergeben sich folgende Eigenschaften (Attribute) eines Personen-Objekts: Vorname, Nachname, eventuell ein Titel und ein Geburtsdatum.

Einem Personen-Objekt können kein, ein oder mehrere Adress-Objekte zugeordnet sein, die aus einer Postleitzahl, einem Ortsnamen, einem Straßennamen und einer Hausnummer bestehen.

Ebenfalls können kein, ein oder auch mehrere Telefon-Objekte einem Personen-Objekt zugeordnet sein. Ein Telefon-Objekt enthält eine Vorwahl und eine Rufnummer.

Für alle Objekte gilt, dass die Bestandsdaten geändert werden können. Außerdem sollen jederzeit neue Objekte hinzugefügt werden können.

Klassen

Alle Personen-, Adress- und Telefon-Objekte zeigen den gleichen Aufbau. Auch die Verhaltensweisen sind für alle diese Objekte gleich. In einem Abstraktionsschritt wird nun aufgrund der Objekt-Eigenschaften und -Verhaltensweisen eine Klasse modelliert.

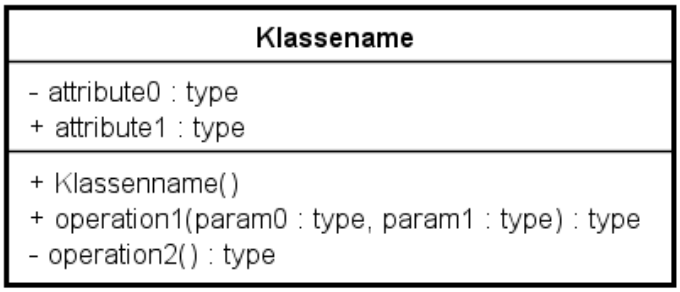
Eine Klasse ist somit ein „Bauplan“ oder eine „Schablone“ zur Erstellung von Objekten des betreffenden Typs!

© Uwe Homm Version vom 11. Oktober 2018

Eine Klasse wird im Rahmen der *„Unified Modeling Language“* (UML) in Form eines Klassendiagramms dargestellt. Ein Klassendiagramm enthält eine oder mehrere Klassen sowie eventuell vorhandene Beziehungen zwischen den Klassen, die sich aufgrund der Objekt-Beziehungen ergeben.

Klassendiagramm

Eine Klasse wird in der UML prinzipiell wie nebenstehend dargestellt beschrieben:



In einem vertikal dreigeteilten Rechteck befindet sich im oberen Teil der Name der Klasse. Der Name einer Klasse wird in C# (und auch in anderen OO-Sprachen) im Singular angegeben und mit einem beginnenden Großbuchstaben geschrieben; z.B. **`Cookie`**.

Sind Klassennamen zusammengesetzt, wird hier die sogenannte CamelCase-Schreibweise[1] verwendet; und zwar die UpperCamelCase-Schreibweise, auch PascalCase genannt. Hierbei wird der erste Buchstabe eines Wortteiles groß geschrieben; z.B. **`ChocolateCookie`**.

Im mittleren Teil werden die Eigenschaften, die alle Objekte dieser Klasse gemeinsam haben, aufgelistet. Für jedes Attribut wird ein Name, ein Datentyp und ein Zugriffsbezeichner angegeben. Attributnamen werden in C# komplett in Kleinbuchstaben geschrieben, wenn es sich um ein einzelnes Wort handelt wie z.B. **`name`** oder **`street`**.

Ist die Eigenschafts-Bezeichnung zusammengesetzt, wird ebenfalls die sogenannte CamelCase-Schreibweise verwendet; jetzt jedoch die lowerCamelCase-Schreibweise, bei der der erste Buchstabe klein geschrieben wird und die nachfolgenden Worte mit einem Großbuchstaben beginnen; z.B. **`myColor`** oder **`nextNumber`**.

Der Typ einer Eigenschaft kann ein grundlegender Datentyp oder auch der Typ einer anderen Klasse sein. Für die Attribute wird ein Zugriffsbezeichner festgelegt: Die beiden wichtigsten sind **`public`** und **`private`**. Der Zugriffsbezeichner **`public`** wird durch ein Plus-Zeichen dargestellt, **`private`** hingegen durch ein Minus-Zeichen.

Im unteren Drittel werden die Methodennamen für die Verhaltensweisen, die diesen Objekten gemeinsam ist, dargestellt. Jede Klasse verfügt über mindestens eine besondere Methode, nämlich eine Konstruktor-Methode. Mit einem Konstruktor wird ein Objekt anhand der Klassenbeschreibung erstellt. Konstruktor-Methoden haben keinen Rückgabewert; auch nicht **`void`**! Eine Klasse kann mehrere Konstruktoren haben, die sich dann in den Parameterlisten unterscheiden müssen. Konstruktoren sind üblicherweise immer **`public`**, damit von außerhalb ein Objekt erstellt werden kann. Mit den Parameter können bei Konstruktoren die zu erstellenden Objekte initialisiert werden.

Andere Methoden können **`public`** (das heißt: von außerhalb des Objektes nutzbar) oder **`private`** (nur innerhalb des Objektes nutzbar) sein. Methoden können Werte eines festgelegten Typs (z.B. **`int`**, **`double`** oder einer anderen Klasse) zurückliefern. Sollten sie keinen Wert zurückliefern, wird als Typ **`void`** angegeben. Des Weiteren können an Methoden Parameter übermittelt werden, die dann innerhalb der Methode nutzbar sind. Diese Parameter haben eine Bezeichnung und einen Typ.

1 Siehe auch **`https://de.wikipedia.org/wiki/Binnenmajuskel#Programmiersprachen`** abgerufen am 12.03.2017

Da die Attribute einer Klasse aufgrund des Geheimnisprinzips üblicherweise mit dem Zugriffsbezeichner **`private`** gekennzeichnet sind und daher kein direkter Zugriff von außen möglich ist, definiert man sogenannte **`get`**- und **`set`**-Methoden. Diese Methoden dienen dazu, einen Attributwert zu lesen (**`Get`**-Methode) oder auch neu zu setzen (**`Set`**-Methode).

Die Methoden werden dann auch entsprechend benannt: Z.B. **`getName()`** oder **`setName()`**.

`Get`-Methoden haben dann als Rückgabe-Datentyp den gleichen Datentyp wie das betreffende Attribut und **`Set`**-Methoden haben einen Parameter mit dem Datentyp des betreffenden Attributs.

Diese Methoden sind wie im Kasten dargestellt sehr einfach aufgebaut.

Methoden werden in C# (abweichend zu anderen OO-Sprachen wie Java oder C++) in PascalCase geschrieben.

```
class Person {
     private string name;
     …
     public string GetName(void) {
     return name;
     }
     public void SetName(string n) {
     name = n;
     }
…
}
```

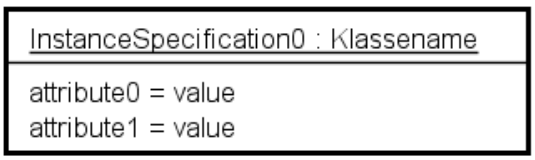
Ebenso werden die unten genannten Properties in PascalCase geschrieben.

Objektdiagramm

Objektdiagramme zeigen wie bei einem Blitzlicht den momentanen Status eines Objektes. Das heißt in einem Objektdiagramm werden für Objekte deren Objektnamen und deren Datentyp (d.i. die Klassenbezeichnung) und die Werte der Attribute angegeben.

Nebenstehend ist ein Objekt in allgemeiner Form dargestellt.

Hierbei stellt „InstanceSpecification0“ den Namen des Objektes dar. Dieser und die durch Doppelpunkt getrennte Klassenbezeichnung werden unterstrichen.



Ein Objekt hat zur Programmlaufzeit in seinen Attributen bestimmte Werte gespeichert, die entsprechend angegeben sind.

Aufgaben

1. Zeichnen Sie für alle im Einführungstext genannten Objekte ein jeweiliges Klassendiagramm mit den genannten Attributen und geeigneten Datentypen sowie einem Konstruktor, der die Objekte initialisiert. Die Beziehungen zwischen den Klassen **`Person`**, **`Telefon`** und **`Adresse`** sind vorerst irrelevant!

2. Informieren Sie sich über die Darstellung von Objektdiagrammen, bei denen Beziehungen zwischen Objekten bestehen.

3. Zeichnen Sie ein Objektdiagramm für die genannten Situation im Beschreibungstext. Verwenden Sie beispielsweise Objektbezeichnungen wie **`p1`** oder **`a1`** oder **`t1`**.

4. In C# gibt es sogenannte Properties, die den Zugriff auf Attribute ermöglichen und eine andere Form der **`set`**- und **`get`**-Methoden sind. Informieren Sie sich über den Syntax. Stellen Sie den Zugriff auf die Attribute in den Klassen auf Properties um.

© Uwe Homm Version vom 11. Oktober 2018