



Programmieren Sie objektorientiert

# Oh, Oh, OOP

Wer heute **nicht objektorientiert** denken kann, kann als Programmierer **einpacken**. Die wesentlichen **Aspekte** der objektorientierten Programmierung sind aber nicht schwer **zu begreifen**.

ULRICH ROHDE

**M**oderne Programme, gerade solche, welche Fenster bedienen, bestehen aus so vielen Einzelaktionen und Datenmanipulationen, dass ohne geeignete Hilfsmittel der Überblick nicht mehr zu behalten ist. Deshalb haben die professionellen Programmierer lange nach Methoden gesucht, mit denen man die Einzelheiten im Griff behalten kann und doch das Ganze nicht aus den Augen verliert. Das Ergebnis ist die objektorientierte Programmiermethode.

## Objekte im Computer

Die ersten Methoden gegen Verwirrung und Spaghetti-Programm-Töpfe waren zunächst die Strukturierte Programmierung und dann die Modulare Programmierung – bis sich einige kluge Köpfe (zum Beispiel die Erfinder von Smalltalk etwa 1970 und einige KI-Leute) etwas einfallen ließen, das mehrere Fliegen mit einer Klappe erschlug. Man begann Computerprogramme als Objekte zusehen, die wie Lebewesen Nachrichten empfangen, verarbeiten und ausgeben konnten, auch an andere Programme im selben Computer. Die Objekte blieben dabei autark und von außen nur über definierte Schnittstellen beeinflussbar. Zuerst nur für KI-Forscher und GUI-Entwickler interessant, entdeckten die Programmiertheoretiker, dass diese Sichtweise große Vorteile hat. Zum einen konnten solche Objekte so programmiert werden, dass sie

ungebetene Eingriffe (Seiteneffekte) von außen nicht zuließen und zum anderen im Inneren so modelliert werden, dass sie eindeutige Verhaltensweisen zeigten. Denn schließlich soll ein Computerprogramm genau das tun, was man will – und nichts anderes.

## Klassen als Schablonen für Objekte

Wie es Lebewesen verschiedener Arten gibt und die Angehörigen einer bestimmten Art sich gleichsam nur durch die verschiedenen Ausprägungen ihrer gemeinsamen Eigenschaften und Fähigkeiten unterscheiden, lassen sich auch Programm-Objekte in Klassen einteilen. Eine Klasse ist gewissermaßen nur die Konstruktionsvorschrift für Objekte derselben Art. In einer Klasse sind die Datenelemente und die zugehörigen Aktionen (sie heißen hier Methoden) zusammengefasst, aus denen ein Objekt bestehen soll. Ein konkretes Objekt entsteht, wenn im Computerspeicher entsprechend der Konstruktionsvorschrift der Klasse durch den Compiler und das Laufzeitsystem Platz für die Daten und Zugang zu den Methoden geschaffen wird. Ein solches Objekt heißt dann eine Instanzvariable (oder nur Instanz) der Klasse. Zwei Instanzvariablen, zum Beispiel Bankkunde1 und Bankkunde2, unterscheiden sich im internen Aufbau nicht, wohl aber in den inneren Zuständen der zugehörigen Bankkonten. Instanzen einer Klasse hören auf dieselben Botschaften und haben dieselbe Funktionalität. Eine Instanz wird durch einen Codeabschnitt initialisiert, der in

der zugehörigen Klassendefinition ausprogrammiert ist oder auf den dort verwiesen wird.

Zusammengefasst: Das Zusammenfügen von Daten mit den Funktionen, die diese Daten bearbeiten, heißt Kapselung. Eine durch Kapselung entstandene Datenstruktur heißt Klasse. Eine Variable oder Konstante, die diese Datenstruktur nutzt, heißt Instanz dieser Klasse und ist ein Objekt. In Objektorientierten Programmiersprachen ist diese Kapselung so realisiert, dass man von außen die Daten einer Klasseninstanz nur über die Methoden der Klasse beeinflussen kann.

## Was ist an Klassen so klasse?

Drei grundlegende Eigenschaften ergeben das Vorteilhafte am Konzept der Klassen:

Kapselung, Vererbung und Polymorphie.

- Das Schöne an der Kapselung, wie sie in OOP-Sprachen realisiert ist, ergibt sich aus der Implementierung der Zugriffsmechanismen. Die Daten in einer Instanz einer Klasse können von außen nicht direkt manipuliert werden, sie sind verborgen. In Non-OOP-Sprachen war die unbeschränkte Datenzugriffsmöglichkeit eine Fehlerquelle ersten Ranges (Seiteneffekte!). Klassen haben ein Innen und ein Außen. Bei der Programmierung einer Klasse kann man festlegen, welche Methoden von Außen zu sehen (aufrufbar) sind, meist mit dem Schlüsselwort "public" deklariert, während nur intern nutzbare Methoden mit "private" deklariert werden.

- Um nicht jedes Mal eine komplette neue Klasse programmieren zu müssen, wenn es um ähnliche Aufgaben geht, gibt es die Vererbung. Das ist ein Mechanismus, mit dem man angeben kann, dass eine Klasse Elemente und Methoden einer anderen Klasse im vorhinein enthalten soll, also erben soll. Die neue



Klasse enthält alle Datenstrukturen und Funktionen der alten Klasse und zusätzlich neue Elemente. Damit wird eine Beziehung zwischen den Klassen realisiert. Die alte Klasse nennt man "Basisklasse", die neue "abgeleitete Klasse". Die Struktur, die entsteht, wenn man ein ganzes System solcher Klassen hat, nennt man Klassenhierarchie.

- Oft könnte man in einer abgeleiteten Klasse eine von der Basisklasse ererbte Methode zwar dem Namen nach, aber nicht der Funktion nach gebrauchen. Dann ist es erlaubt diese abgeleitete Methode neu zu programmieren. Es entstehen zwei verschiedene Funktionen, die den gleichen Namen haben, aber doch auseinander zu halten sind, weil sie in verschiedenen Klassen residieren. Dieser Sachverhalt heißt Polymorphismus.

### Ein kleines Beispiel für Klassen

Jeder Programmierer muss irgendwann einen Zähler gebaut haben. Das Schema, den Typ "Zähler" könnte man freiweg in einer objektorientierten Fantasiensprache so notieren:

```
Klasse Zähler:
int Stand;
public int pluseins();
public int minuseins();
public int showStand();
```

Diese Struktur aus einer Integer-Variablen und drei öffentlichen Funktionen wird im Speicher unter dem Namen "counter" aufgebaut, wenn der Compiler unserer Fantasiensprache die Zeile

```
new counter Zähler
```

erreicht. Gleichzeitig kann bei dieser Konstruktion (die ein Konstruktor durchführt) noch Code zusätzlich ausgeführt werden, der den Zähler initialisiert. Er muss bei der Klassenvereinbarung programmiert werden und heißt dort ebenfalls Konstruktor. Die drei Funktionen müssen auch bei der Vereinbarung der Klasse Zähler programmiert worden sein, damit die Instanzen auch zählen können.

Ein anderes Programm kann mit

```
counter.pluseins();
```

### IM INTERNET GEFUNDEN

C: God's programming's language  
If true then C++ is a realization of  
the mistakes God made with it.

den Zähler inkrementieren, weil "pluseins()" öffentlich zugänglich ist.

Es gibt Leute, die nur bis drei zählen wollen. Für die sei hier eine entsprechende Klasse in der Fantasiensprache vereinbart:

```
Klasse ZählerBisDrei: beerbt Zähler
(alles Zählerelemente stehen jetzt
zur Verfügung)
private int beidreireinnullse-
tzen();
```

Hier muss der Code, der zur neuen Funktion "beidreireinnullsetzen()" dazu kommt die Variable "Stand" laufend beobachten und sie auf Null setzen, sobald der Wert 3 erreicht ist.

```
new ModuloDrei ZählerBis
Drei
```

ruft die Instanz "ModuloDrei" ins Leben, die zum Beispiel "pluseins()" als ererbte Methode besitzt und bei der nach dreimal Aufruf von

```
ModuloDrei.pluseins();
```

die Ausgabe

```
ModuloDrei.showStand();
```

die Zahl 2 ausgegeben würde. Und nach dem vierten Mal "modulodrei.pluseins()" würde die private Methode "beidreireinnullsetzen()" in "ModuloDrei" sicherstellen, dass danach "ModuloDrei.showStand()" Null ausgibt.

Der Programmierer ist es, der die Klassen entwerfen muss. Was daran zunächst umständliche Schreibarbeit zu sein scheint, wird in großen Programmsystemen zum Vorteil, weil über die Klassen die Objekte unmissverständlich definiert sind, Code-Wiederverwendung bei kluger Wahl der Klassen leicht und durchschaubar gemacht wird und eine klare Hierarchie im Computer entsteht. Die Wahl der Klassen muß klug sein; man kann auch objektorientiert schlecht programmieren. So haben manche Programmiersprachen die abstrakte Klasse Objekt definiert, die den Rahmen zur Ableitung sämtlicher Klassen bereitstellt, indem sie einen virtuellen Konstruktor zur Verfügung stellt, der in abgeleiteten Klassen ausprogrammiert werden muss und deren Instanzen einrichten hilft. Ein passend programmierter Destruktor in einer Instanz sorgt für die Speicherbereinigung, wenn ein Objekt nicht mehr benötigt wird. Das alles werden Sie auf den nächsten Seiten in dieser Ausgabe noch oft vorgeführt bekommen. Viel Nutzen daran.