

Die drei wichtigsten Grundlagen

Was ist denn objektorientierten Programmierung (OOP) nun genau? Wie in der Einführung schon dargestellt, ist es einer der zentralen Aspekte der OOP, Daten mit den zugehörigen Funktionen zusammenzuführen. Diesen Prozeß bezeichnet man als *Kapselung*, die daraus resultierende Datenstruktur nennt sich *Klasse*, eine Variable oder Konstante, die diese Datenstruktur nutzt, nennt sich eine *Instanz* der Klasse. Der wesentliche Unterschied einer Klasse des OOP zu den schon bekannten Datenstrukturen (z.B. ein `struct` in C) ist, daß eine Klasse zusätzlich zu den Daten auch weitreichende Funktionalitäten übernehmen kann. Dadurch wird sie zu einem *Abstrakten Datentyp*, der zu den Inhalten (den Daten) auch die Möglichkeiten spezifiziert, mit ihnen umzugehen (die Funktionen). Die Daten einer Klasse werden dabei als *Elemente*, und die Funktionen als *Elementfunktionen* oder *Methoden* bezeichnet.

Der zweite wichtige Aspekt der OOP ist die Möglichkeit der *Vererbung*. Um nicht jedesmal eine komplette neue Klasse programmieren zu müssen, kann man in den objektorientierten Sprachen angeben, daß eine Klasse alle Elemente und Methoden einer anderen Klasse 'erben' soll, d.h. die neue Klasse enthält von vorneherein alle Daten und Funktionen der alten Klasse. Diese können dann um weitere ergänzt werden oder bei Bedarf auch überschrieben werden. Damit wird eine Beziehung wie z.B. 'ein Apfel ist eine Frucht' realisiert, durch zusätzliche Elemente und Methoden wird die neue Klasse genauer spezifiziert. Die alte Klasse wird dabei als *Basisklasse* bezeichnet, die neue als *abgeleitete Klasse* und die Struktur, die durch die Vererbung entsteht, als *Klassenhierarchie*.

Der dritte grundlegende Aspekt ist der des *Polymorphismus*. Wie erwähnt kann man in einer abgeleiteten Klasse Methoden der Basisklasse überschreiben. Dadurch hat die abgeleitete Klasse eine neue Methode, die aber immer noch den gleichen Namen trägt wie die Methode der Basisklasse. Das Prinzip des Polymorphismus besagt nun, daß zur Compilezeit noch nicht feststehen muß, welcher Klasse einer Hierarchie ein Datum angehört, solange gewährleistet ist, daß die benutzten Elemente und Methoden in allen möglichen Klassen vorhanden sind. Welche Methoden dabei aufgerufen werden, entscheidet sich dann nicht wie sonst üblich zur Compilezeit, sondern erst zur Laufzeit. Dies bezeichnet man auch als *späte* oder *dynamische Bindung* - im Gegensatz zur herkömmlichen *statischen Bindung*.