

Objekt orientiertes programmieren

Von:

Heinz – Volker Viehof

Warum eine Abhandlung Über „OOP“ ?

Weil Ich bisher noch keine vernünftige Erklärung gefunden habe was „Objekt-Orientierte Programmierung“ eigentlich ist!

Zuerst :

Hier werde Ich die Grundzüge der objekt-orientierten Programmierung anhand von Beispielen aus dem täglichen Leben und an Beispielen kommerzieller Programme Erläutern. Hierbei stütze ich mich auf meine eigenen Erfahrungen und nicht auf Sekundär Literatur. Ich kann auch leider keine gute zu dem Thema empfehlen.

In diesem Text werden die genannten Beispiele oft zur Erklärung nützlicher sein als Die Beschreibung des Sachverhalts.

Vorraussetzungen zum Verständnis sind Kenntnisse des Windows© Betriebssystems und eines (Echtzeit) Strategiespiels z.B.: Command & Conquer© oder die Siedler©.

Wichtig zum Verständnis des Konzepts ist ein „Gefühl“ dafür zu bekommen, was Objekte, Klassen und Instanzen sind.

Im Endeffekt wird man feststellen, das die Zusammenhänge wesentlich einfacher bzw. trivialer sind als sie zuerst Aussehen!

Was bedeutet „Objekt-orientiert“?

Objekt orientiertes programmieren ist eine Programmierstrategie, es setzt keine spezielle Programmiersprache voraus, jedoch unterstützen bestimmte Sprachen z.B. Python das Objekt orientierte Programmieren sehr gut.

Objekt orientiert heißt, das Daten, Programme, Threads, Routinen und so weiter als (Daten-)Objekte betrachtet werden. Diese Betrachtung von Software bzw. Daten ist ganz konform mit den Ideen von John von Neumann und Konrad Zuse.

Die Vorteile :

Programmteile oder Daten oder Funktionsdefinitionen brauchen nur einmal erstellt zu Werden, und können von anderen, evtl. vollständig voneinander unabhängigen Programmen genutzt werden (z.B.: benutzt jedes Programm unter Windows die in der „Win.ini“ Voreingestellte CodePage, Landessprache oder Maßsystem)

Begriffsdefinitionen mit Beispielen :

- Klasse -

Die Gesamtheit aller Objekte

Alltagsbeispiel:

die Säugetiere sind eine Unterklasse der Klasse Tiere

Windows :

die Klasse der <Gerätetreiber> oder die Klasse <Programme>

Strategiespiel :

die Klassen <Gebäude> ,<Fahrzeuge> oder <Personen>

- Unterklasse -

Eine Klasse innerhalb einer Klasse

Alltagsbeispiel:

die Menschen sind eine Unterklasse der Klasse Säugetiere

Windows:

<Ordner> sind eine Unterklasse von <Daten>

Strategiespiel:

die Unterklassen <Zivilgebäude> ,<Industrie> oder <Verteidigungseinrichtung> der Klasse <Gebäude>

- Objekt -

Ein Objekt ist eine Instanz einer Klasse oder Unterklasse

Alltagsbeispiel:

eine Person ist ein Objekt der Klasse Mensch

Windows:

<explore.exe> ist ein Objekt der Klasse <Programme> , <Win.ini> ist ein Objekt der Klasse <Daten>

Strategiespiel:

ein einzelnes Fahrzeug z.B.: <Panzer> ist ein Objekt der Klasse <Fahrzeug>

- Instanzieren, Instanz -

Ableiten des Objekts aus der Klasse, das Objekt ist dann die Instanz einer Klasse.

Rufe ich unter Windows ein Programm mehrfach auf, so instanziiere ich das Programm, jedes offene Programmfenster (Objekt) ist eine Instanz des Programms z.B.: mehrere Downloads verschiedener Dateien gleichzeitig

Alltagsbeispiel:

Jede <Fahrkarte> ist eine Instanz (und damit ein Objekt) der Klasse <Fahrscheine>

Windows:

Klick den Internet Explorer mehrmals an, Du hast dann Verschiedene Instanzen des IExplorers und kannst gleichzeitig vollkommen verschiedene WebSites besuchen.

(macht nicht wirklich Sinn) ,aber Du kannst zum Beispiel Word zweimal öffnen um an zwei verschiedenen Dokumenten zu arbeiten.

Strategiespiel:

Produziert die Waffenfabrik <Panzer> so bildet sie neue Instanzen der Klasse <Panzer>.

- Attribute -

Jedes Objekt hat Attribute die seine Eigenschaften umschreiben, auch Klassen und Unterklassen haben Attribute

Alltagsbeispiel:

<Höchstgeschwindigkeit> oder <Leistung [kW]> sind Attribute des Objekts <Auto>

<Haarfarbe>, <Name>, <Größe> sind Attribute des Objekts <Person>

<aufrechter Gang> ist ein Attribut der Klasse <Mensch>

Windows:

<nur Lesen> ist ein Attribut von z.B.: <Readme.doc>

Strategiespiel:

ein Objekt z.B.: < Panzer> hat die Attribute <Geschwindigkeit>, <Richtung>, <Bewaffnung>, <Zustand> -

Alle diese Attribute beschreiben das Objekt so das Voraussagen über das weitere Verhalten und Schicksal des Objekts getroffen werden können.

- Methoden -

Können auf ein Objekt (oder Klasse oder Unterklasse) angewandt werden, bzw. werden auf Objekte, Klassen oder Unterklassen angewandt. Wird eine Methode an einem Objekt angewandt so ändert dies in der Regel die Attribute des Objekts. Wird eine Methode an einer Klasse oder Unterklasse angewandt so ändert dies die Attribute der zukünftig Instanziierten Objekte der Klasse.

Alltagsbeispiel:

man kann das Objekt Auto starten, bzw. die Methode <starten> oder <beschleunigen> auf das Objekt <Auto> anwenden,

aber auch z.B.: <verkaufen>, <reparieren> oder <verschrotten>, wendet z.B.: BMW die Methode <Preissenkung> auf Klasse <5er> an,

so betrifft dies alle von da an gebauten Fahrzeuge, das Attribut <Verkaufspreis> wurde geändert.

man kann die Methoden <kommunizieren> oder <ignorieren> auf das Objekt <Person> anwenden.

Windows:

Methode <Löschen> am Objekt <Spam.doc> anwenden, Methode <Download> am Objekt <Freeware.zip> anwenden.

(wendet man eine Methode z.B.: <Kopieren> an einem ganzen Ordner an, so wendet man die Methode natürlich am Objekt <Ordner> und an der Klasse <Objekte im Ordner> an!)

Strategiespiel:

hier gilt genau das oben gesagte Ich kann z.B.: auf die Unterklasse <Panzer> die Methode <Bauen> anwenden, auf das Objekt <Panzer> die Methoden <reparieren>, <fahren> oder <schießen> anwenden.

- Vererbung -

Es werden die Attribute von der Klasse an die Unterklasse und von den Unterklassen an die Objekte vererbt.

Alltagsbeispiel:

Die Klasse <Kraftfahrzeuge> hat das Attribut <Energieverbrauch>, die Unterklasse <PKW> besitzt auch das Attribut <Energieverbrauch>, die Unterklasse <3er BMW> besitzt ebenso das Attribut <Energieverbrauch> und das Objekt

<3er BMW, Fzg. Nr.:BMW XXX 00815007,Besitzer Fr. Mustermann> natürlich auch !!!

Strategiespiel:

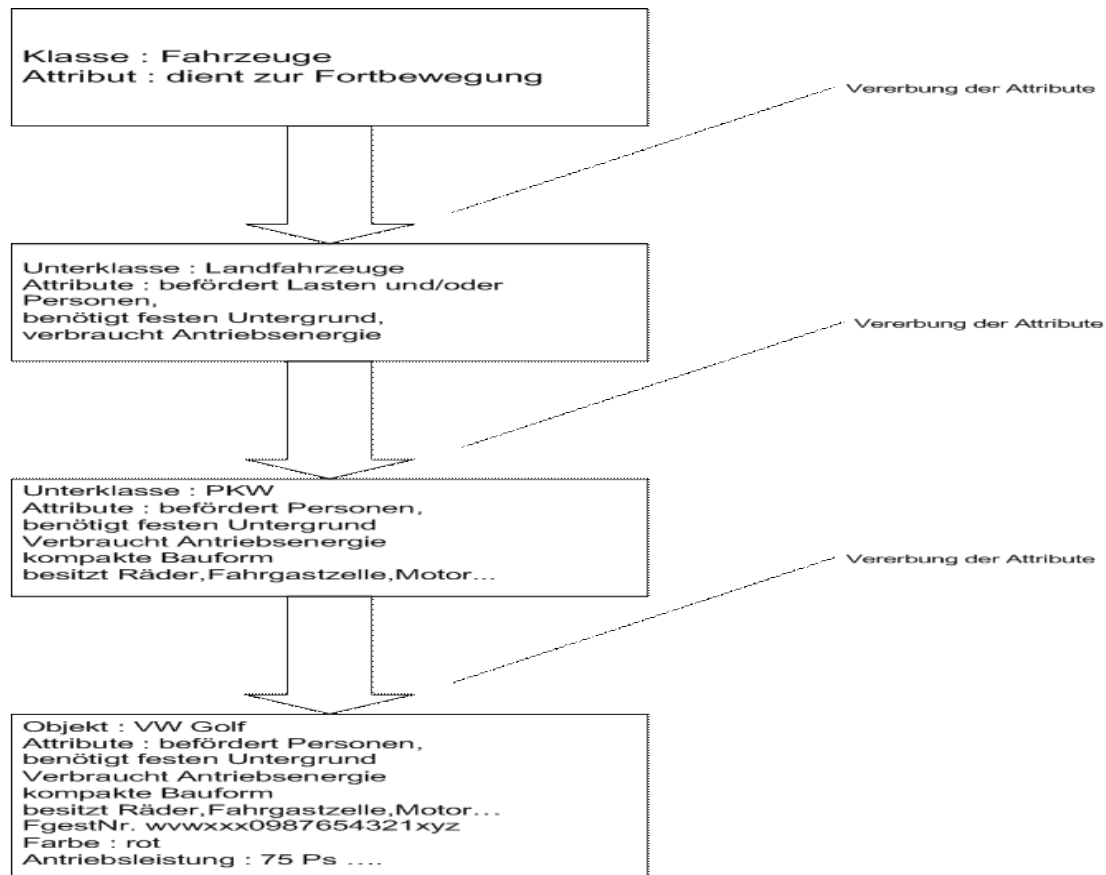
Die Klasse <Personen> besitzt das Attribut <schwach gepanzert>, die Unterklasse <MG Schütze> ist schwach gepanzert, und jede auf dem Bildschirm sichtbare <Person> selbstverständlich auch.

Konstruktoren und Destruktoren

Konstruktoren sind Methoden zum Erschaffen neuer Instanzen, und Destruktoren zerstören diese.

Ein konkretes Beispiel wäre zum Beispiel der Aufruf 'CreateWindowsEx' in C++ oder Assembler, der ein neues Fenster, das ja auch ein Objekt darstellt erschafft. Dieses Fenster, bzw. diese Instanz von 'Fenster' besitzt ein individuelles Handle, das der Konstruktor als Rückgabewert übergibt. Analog dazu 'zerstört' der Destruktor 'ExitWindowsEx' das Fenster das über das Handle spezifiziert wird. Wie aus dem Prinzip der Vererbung hervorgeht, betrifft dieses auch alle 'Childwindows' die Vorher vom Hauptfenster erzeugt wurden.

Schematische Darstellung des Prinzips {Klasse; Unterklasse; Objekt}



Anmerkung: Die Nennung von Produkt und Markenbezeichnungen im Text (Software- und Autofirmen bzw. ihrer Produkte) hat den Zweck die Beispiele authentischer zu halten, es ist keinesfalls als Werbung aufzufassen !

Oop/Html-version/hvviehof/26.08.2004

Geändert und ergänzt am 14.09.2004, 13.10.2007