

## 7 OOP- Grundlagen

OOP ist vor allem bei großen Programmen sehr hilfreich, da ein Objekt sehr leicht eingebunden oder verändert werden kann. Dies verbilligt vor allem große Softwareprojekte enorm, da der Wartungsaufwand erheblich reduziert werden kann.

Was ist ein Objekt überhaupt? Ein Objekt hat Eigenschaften, die dieses Objekt einzigartig machen

D.h. Ein Computer beispielsweise wäre ein Objekt, er hat einen Prozessortyp, einen Bus-typ, eine Gehäuseart,... alle diese Attribute machen ihn (das Objekt) einzigartig.

Es stellt sich die Frage, wie die Daten zu speichern sind. Die beste Lösung ist eine Struktur, in der alle Daten, das Objekt betreffend gespeichert werden. (1.Schritt in Richtung OOP)

Ein Objekt hat also eigene interne Daten (Instanz Variablen) und außerdem eigene Funktionen (Methoden), die nur für dieses Objekt gelten. D.h. Ein Objekt ist einfach eine Struktur in die noch ihre Funktionen direkt hinein geschrieben werden. (2. Schritt in Richtung OOP)

Damit das Programm weiß, wie auf die einzelnen Daten zugegriffen werden darf gibt es mehrere reservierte Wörter. Diese sind u.a. private, public, protected

public: dieses Wort sagt aus, daß überall aus dem Programm, und das jedes Objekt auf die Daten die mit private definiert sind zugreifen darf.

private: dies bedeutet, daß nur objekteneigene Funktionen auf diesen Datenbestand zugreifen darf.

protected: bei dieser Definition wird festgelegt, daß nur objekteneigene Funktionen und Funktionen von Vererbungen dieses Objektes auf diese Daten zugreifen dürfen.

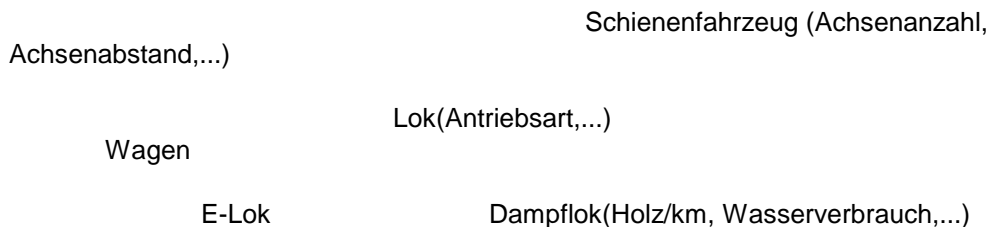
So ein Unterprogrammaufruf wird beim OOP auch als schicken einer Nachricht bezeichnet.

Wir rufen eine Methode auf, indem wir das Objekt nennen einen Punkt setzen und die gewünschte Methode angeben.

z.B. Rechteck.füllen

Bei Strukturen ist die Gefahr, daß der Programmierer auf Datenbestände zugreift, die er zu diesem Zeitpunkt vielleicht gar nicht ändern können sollte, daher Datenkapselung (private, public, protected siehe später). D.h. die dafür definierten Funktionen ändern die Datenbestände eines Objektes (und sonst nichts, nur diese Funktionen).

Beispiel zu Klassen: Einteilen von Zügen



Oben wird eine Hierarchie von Objekten angeführt. D..h in Lok muß keine Achsenanzahl mehr gespeichert werden, weil Lok ein Schienenfahrzeug ist und dort die Information zu Achsenzahl gespeichert wird, Jedes Schienenfahrzeug muß demzufolge Achsen haben um als Schienenfahrzeug zu gelten. Wasserverbrauch wiederum wird nur bei Dampflok gespeichert, da eine E-Lok keinen Holzverbrauch aufweist.

## 8 Vererbung (Inheritance)

Viele Objekte sind einander ähnlich, jedoch nicht ident. Um nicht jedes Objekt extra definieren zu müssen Vererbung. Man muß nur sagen, welches Objekt von welchem O. die Daten und Methoden erben soll und welche Daten und Funktionen zusätzlich enthalten sind.

Siehe Beispiel mit Lokomotiven von vorhin.

Wir vererben die Datenbestände und die Funktionen und könne weitere hinzufügen.  
Wir kennen Basis (Parent, Eltern)- Klassen und Sub (Child, Kinder) Klassen

Eine Virtual Funktion ist ein Platzhalter, der festlegt, daß nach der Vererbung unbeding z.B. eine Funktion an dieser Stelle geschriben werden muß.

## 9 Dynamisches Binden und Polymorphismus

Dynamischs binden

Objekte sind dynamisch, d.h. sie werden erst bei laufendem Programm erzeugt (constructor), bzw. gelöscht (destructor). Dies ist in etwa mit dem Speicherallocieren in C „malloc“, welches letztes Jahr im Unterricht durchgenommen wurde vergleichbar. Dies bringt den erheblichen Vorteil, daß beim Compilieren noch nicht genau feststehen muß an welche Stelle (im Assembler-code) gesprungen werden muß. An unklaren Stellen wird ein Platzhalter eingesetzt, der während der Laufzeit durch das Run-time Modul richtig interpretiert wird.

Anders gesagt bedeutet „D.B.“: es muß bei Compilierung noch nicht feststehen, welches UP ausgeführt wird

Z.B. bei Benutzerabhängigen Eingabe. Zeigt Pointer auf O1 oder O2

z.B. printf <> cout

Polymorphie (Polymorphe arrays)

Bei einem normalen Feld können nur Daten gleichen Typs gespeichert werden (int). Da jedoch bei Objekten Pointer verwendet werden können verschiedenste Objekte (bzw. Pointer auf diese) in einem Array gespeichert werden.

Auf den ersten Blick sieht es so aus als würden verschiedenste Objekte in einem Feld gespeichert werden können (z.B. verschiedene Eisenbahnen, Züge und Wagen) in Wirklichkeit wird jedoch jeweils nur ein Pointer auf ein Objekt gespeichert und dadurch wird die polymorphie (vielschichtigkeit, vielgesichtigkeit) ermöglicht.

Garbage Collection

D.H. Alle Objekte auf die kein Pointer zeigt werden aus dem Speicher gelöscht.