26 Einblick in objektorientierte Programmierung OOP

Dieses Kapitel 26 mit OOP war mal angedacht, aber die Übererfüllung des Lehrverpflichtung eines Professors des Freistaates Bayen vor über zwei Jahrzehnten mit über 65 Lebensjahren standen einer Überarbeitung des Kapitels im Wege. Und nur auf Wunsch eines Besuchers meiner Homepage www.karl-haller.de im Oktober 2025 gebe ich diese Seite mit großem Vorbehalt frei.

😊 kha, 30.10.2025

Gliederung

26.1 Vorbemerkungen

Objektorientiertes Programmieren (Abkürzung OOP) ist eine moderne Programmiertechnik, die zwar in der Anwendung der natürlichen Denkweise eher entspricht, aber einen gänzlich anders gearteten Aufsatz auf die bisherige prozedurale Programmiertechnik bedeutet. OOP ist zwar erst gegen Ende der achtziger Jahre allgemein bekannt geworden, die Grundlagen dazu sind aber schon 10 Jahre früher geschaffen worden. OOP verfolgt das Ziel, Problemlösungen nicht immer neu zu entwickeln, sondern bereits vorhandene Quellen zu nutzen.

OOP setzt geeignete Programmiersprachen voraus. Man unterscheidet zwischen ausschließlichen OOP-Sprachen (bekanntestes Beispiel ist Smalltalk) und Hybridsprachen, das sind konventionelle Programmiersprachen mit einem OOP-Aufsatz, der nach dem Ermessen des Programmierers genutzt werden kann oder nicht. Turbo-Pascal hat in der Version 5.5 die ersten OOP-Elemente als Aufsatz bekommen, der in den Nachfolge-Versionen weiter ausgebaut wurde. C-Compiler mit OOP-Aufsatz tragen die Zusatzbezeichnung "++", haben also den Namen "C++", z.B. von Microsoft oder Borland. Im weitesten Sinne kann man auch der Seitenbeschreibungssprache PostScript OOP-Eigenschaften zusprechen.

Objektorientierte Programmierung setzt die Existenz der benötigten Objekte voraus. Diese müssen also vorab mit den Mitteln der herkömmlichen Programmiertechnik (Prozeduren und Funktionen) geschaffen oder aus Bibliotheken, z.B. Turbo-Pascal-Units, übernommen werden. Im Normalfall geht der Programmierer somit von bereits vorhandenen Objekten aus. Der Objektinhalt ist nach außen hin weitgehend "unsichtbar", dem Programmierer müssen aber die im Objekt verankerten Namen der **Methoden** (Prozeduren und Funktionen) bekannt sein

26.2 Das Konzept der objektorientierten Programmierung

Das Konzept der OOP umfaßt die Begriffe

a) Datenkapselung,
b) Vererbung,
c) Polymorphismus.

Zu a): Datenkapselung:

Die Zusammenfassung von Daten und Routinen (genauer Methoden) zu der Einheit Objekttyp (auch Klasse genannt) wird Datenkapselung genannt.

Sie hat in Pascal das Format:

```
type
  objektyp = object
               [datenliste]
               [methodenliste]
             end;
```

Die eckigen Klammern der Formatbeschreibung sind wie üblich Meta-Symbole und stehen für "optional". Kursiv gesetzte Bezeichnungen stellvertreten - wie in diesem Skriptum üblich - benutzerspezifische Eintragungen.

Es ist gerade bei OOP üblich, aber nicht zwingend, den Bezeichner des Objekttypen mit einem großen "T" beginnen zu lassen.

Im Falle der erst anschließend behandelten *Vererbung* lautet das Format:

```
type
  objektyp =   object(vorfahre)
               [datenliste]
               [methodenliste]
             end;
```

In Turbo-Pascal ist "object" ein reserviertes Wort, kann also nicht umdefiniert werden.

Die Deklaration eines Objekts, auch Instanz eines Objekttyps genannt, selbst erfolgt nach der Typ-Deklaration in konventioneller Weise in der Variablendeklaration mit:

```
var
  objektbezeichner: objekttyp;
```

Die "Datenliste" enthält Variablendeklarationen, allerdings ohne ein vorausgestelltes "var" und gleicht somit der Deklaration eines Rekord-Typs. Diese Variablen sind zum betreffenden Objekttyp lokal, d.h. auf sie kann nur von den Methoden des gleichen Objekttyps oder davon abgeleiteten Objekttypen zugegriffen werden.

Die "Methodenliste" enthält die Namen der Methoden (Routinen), ggf. mit formalen Parametern, nicht aber die Implementierung der Methode selbst. Die Methodenbezeich-

ner sind ebenfalls lokal zum betreffenden Objekttyp, so daß auf sie auch nur vom betreffenden Objekttyp selbst oder davon abgeleiteten Objekttypen zugegriffen werden kann.

Eine Botschaft an ein Objekt hat das Format:

```
objektbezeichner.methodenbezeichner;   { ohne aktuelle Parameter }
```

bzw.

```
objektbezeichner.methodenbezeichner(aktuelleParameter, ...);
```

Man beachte den Punkt zwischen Objektbezeichner und Methodenbezeichner!

Die Methoden selbst können **statisch** oder **virtuell** sein.

Bei einer **statischen** Methode wird die Methode bereits nach dem Kompilieren in das aktuelle Programm eingebunden (früher Bindung).

Bei einer **virtuellen** (= dynamischen) Methode wird die Methode erst bei der Laufzeit in das aktuelle Programm eingebunden (späte Bindung). Bei einer virtuellen Methode muß in der Methodenliste nach dem Bezeichner der Methode und der optionalen Liste der formalen Parameter das reservierte Wort "**virtual**" stehen, eingeschlossen in Semikolons. Virtuelle Methoden besitzen eine Reihe von Vorteilen, die vor allem bei komplexen Programmen zum Tragen kommen. Nachteilig ist die etwas geringere Ausführungsgeschwindigkeit.

Wenn eine virtuelle Methode deklariert ist, müssen alle abgeleiteten Objekttypen folgende Bedingungen erfüllen:

- alle Methode müssen wieder virtuell sein,
- falls eine Parameterliste existiert, muß diese völlig übereinstimmen (d.h. Anzahl, Bezeichner und Datentyp),
- der Objekttyp muß einen Konstruktor zum Initialisieren des Objektes besitzen.

In Turbo-Pascal ist der Konstruktor vergleichbar einer Prozedur, wird aber nicht mit "procedure" sondern mit der reservierten Bezeichnung "**constructor**" aufgerufen. Als Konstruktor-Bezeichner wählt man häufig "Init" oder ähnliches.

Jedes Objekt, das virtuelle Methoden enthält, muß durch den Konstruktor initialisiert werden, ansonsten stürzt das Programm ab, wenn nicht die Compiler-Direktive **{$R+}** gesetzt wurde, was im Entwicklungsstadium unbedingt zu empfehlen ist.

Zu b): Vererbung

Ein wesentliches Merkmal von OOP ist die **Vererbung (Inheritance)**. Damit kann man neue Objekttypen schaffen, die alle Daten und Methoden der Quell-Objekttypen erben und die um neue Daten und Methoden erweitert werden können. Ein Objekttyp kann mehrere Nachkommen, aber nur einen Vorfahren haben. Der Quell-Objekttyp besitzt natürlich keinen Vorfahren.

Zu c: Polymorphismus:

Von Polymorphismus (Vielgestaltigkeit) spricht man, wenn die gleiche Botschaft an Objekte unterschiedlichen Objekttyps unterschiedliche Wirkungen besitzen. Polymorphismus setzt virtuelle Methoden voraus.

26.3 xxxxx

26.4 yyyyy

26.5 zzzzz