KAPITEL 10

OBJEKTORIENTIERTES PROGRAMMIEREN

Das objektorientierte Programmieren setzt sich immer mehr durch. Neue und sich rasch verbreitende Sprachen wie z.B. Java tragen sicherlich dazu bei. Von der Objektorientierung verspricht man sich eine klarere Programmstruktur und letztlich leichtere Wiederverwendbarkeit von bereits geschriebenem Code. Auch Perl (ab Version 5) bietet eine einfache Möglichkeit, objektorientierte Skripte zu erstellen, um von den Vorteilen dieses Programmierparadigmas zu profitieren. Perl stellt jedoch keine klassische objektorientiere Programmiersprache dar, in der objektorientiert programmiert werden muß. Perl überläßt die Wahl, inwieweit Skripte objektorientiert erstellt werden, ganz dem Programmierer, der sein Skript prozedural, objektorientiert oder mit einer Mischung aus beidem erstellen kann.
Dieses Kapitel führt den Leser in die Erstellung objektorientierter Perl-Skripte ein. Für das Verständnis des dargestellten Wissens ist es jedoch sehr nützlich, wenn der Leser bereits die Prinzipien des objektorientierten Programmierens kennt, da diese im Rahmen des Buchs nicht dargestellt werden können.

10.1 BEGRIFFSERLÄUTERUNGEN

Bevor genauer auf die Möglichkeiten des objektorientierten Programmierens in Perl eingegangen wird, sollen zunächst kurz die wichtigsten Begriffe aus diesem Gebiet eingeführt werden:

- Objektklassen
  Objektklassen stellen eine Beschreibung eines Konzept oder eines Gegenstands dar. Sie bestehen aus einem Attributteil, mit dem Eigenschaften des Konzepts oder des Gegenstands beschrieben werden können, und einem Methodenteil, der die möglichen Aktionen für die Klasse und für Objekte der Klasse festlegt. Beispielsweise enthält eine Klasse `rectangle` Attribute (Variablen) zur Beschreibung der Größe, Linien- und Füllfarbe etc. jeden Rechtecks. Darüber hinaus wird es z.B. Methoden geben, die es erlauben, Rechtecke darzustellen, zu verschieben oder deren Farbe zu ändern.

- Objekte, Instanzen
  Ein Objekt oder eine Instanz bezeichnet eine konkrete Ausprägung

einer Klasse. Hierzu werden die in der Klasse angegebenen Attribute mit konkreten Werten belegt. Eine Instanz, beispielsweise einer Klasse `rectangle`, ist daher ein ganz bestimmtes Rechteck, mit einer bestimmten Größe, Linien- und Füllfarbe etc.

- Methoden, Elementfunktionen
  Eine Methode ist nichts anderes als eine Perl-Funktion. In der objektorientierten Programmierung wird der Begriff Methode jedoch vorgezogen. Methoden ermöglichen es, Aktionen auf einer Instanz oder für die Klasse selbst auszuführen. Arbeitet die Methode auf einer Instanz einer Objektklasse, wird diese auch Instanz-Methode genannt. Für die Klasse Rechteck sind beispielsweise Methoden zum Verschieben des Rechtecks und zur Änderung der Farbe Instanz-Methoden, da sie jeweils die Attribute einer Instanz verändern. Instanz-Methoden bilden die Schnittstelle des Objekts zum Rest des Programms. Im Idealfall wird nur über diese Methoden auf das Objekt zugegriffen. Methoden, die unabhängig von einer Instanz sind, werden als Klassen-Methoden bezeichnet. Eine wichtige Klassen-Methode ist der Konstruktor, der zur Erzeugung einer neuen Instanz (z.B. einem Rechteck mit bestimmter Größe und Farbe) einer Klasse benötigt wird. Darüber hinaus dienen Klassen-Methoden zur Ausführung von Aktionen, die alle Objekte einer Klasse betreffen. Beispielsweise kann für die Klasse `rectangle` eine Klassen-Methode definiert werden, die Buch über die Zahl der erzeugten Rechtecke führt, oder eine andere Methode, die durch einen Aufruf alle erzeugten `rectangle`-Objekte auf dem Bildschirm darstellen oder verbergen kann.

10.2 KLASSEN, OBJEKTE UND METHODEN

Die Programmierung von Klassen-Objekten und -Methoden in Perl basiert auf schon bekannten Perl-Elementen. Perl stellt keine besondere Syntax etwa zur Definition einer Klasse bereit.

Eine Klasse wird in Perl durch ein `package` definiert, dessen Name identisch mit dem Namen der Klasse ist. Das Package enthält verschiedene Methoden mit deren Hilfe auf Objekten der Klasse gearbeitet werden kann. Eine Methode wiederum ist in Perl einfach eine Funktion des entsprechenden Packages. Eine Instanz ist in Perl nichts anderes als eine Referenz auf ein anonymes Hash, das einer bestimmten Klasse zugeordnet ist. Das Hash beinhaltet die Daten (die Attribute) für eine Instanz der Klasse.

Grundsätzlich ist das bereits alles, was die objektorientierten Programmierung in Perl ausmacht. Die hier angesprochenen Punkte werden in den folgenden Abschnitten ausführlich dargestellt. Dabei wird deutlich wer-

den, daß vieles in der objektorientierten Programmierung in Perl reine Konvention ist, an die sich sowohl der Programmierer als auch der Anwender einer Klasse halten sollten.

In der Regel wird eine Klasse in eine eigene Datei (mit der Endung `.pm`) geschrieben. Das Skript, das diese Klasse verwenden möchte, kann das Klassen-`package` mit Hilfe der `use`-Anweisung laden. Da in einer Datei beliebig zwischen Packages gewechselt werden kann, ist es jedoch ebenso möglich, daß die Klasse und das diese Klasse verwendende Skript in der selben Datei stehen.

KLASSEN UND OBJEKTE

Eine Klasse in Perl ist einfach ein Package. Dieses Package enthält alle der Klasse zugeordneten Attribute und Methoden. Durch die Verwendung eines Package ist sichergestellt, daß beispielsweise Namen von Methoden verschiedener Klassen nicht in Konflikt miteinander geraten können.

Um ein Objekt einer Klasse erzeugen zu können, enthält jede Klasse einen sogenannten Konstruktor. Der Konstruktor ist eine Funktion, die eine Referenz auf ein neu erzeugtes anonymes Hash (das Objekt) liefert, das dieser Klasse durch den Aufruf der Perl-Funktion `bless` zugeordnet wird. Der Name der Konstruktor-Methode ist per Konvention `new`, da in anderen Sprachen (z.B. C++) dieser Name für den gleichen Zweck eingesetzt wird. In Perl ist der Name dieser Funktion im Prinzip beliebig. Da ein Benutzer dieser Klasse den Namen des Konstruktors jedoch kennen muß, um ein Objekt der entsprechenden Klasse erzeugen zu können, macht es Sinn, sich an diese Konvention zu halten.

In Perl existieren im Gegensatz zu anderen objektorientierten Programmiersprachen keine Destruktor-Funktionen zum Löschen von Objekten. Ein Objekt wird automatisch gelöscht, wenn keine Referenz mehr auf dieses Objekt existiert. Der Programmierer muß sich also auch in diesem Fall keine Gedanken über die Speicherverwaltung für Objekte machen.

Eine einfache Klasse mit dem Namen `anyClass`, die nicht mehr enthält als den Konstruktor, sieht damit wie folgt aus:

```
package anyClass;
sub new {
   my($self);
   $self={};      # Neues anonymes hash erzeugen
   bless $self;   # Dieses Hash der Klasse zuordnen
   return($self); # Referenz als Ergebnis liefern
}
```

Dieses Beispiel verwendet die Variable `$self` zur Speicherung einer Referenz auf das Objekt. Der Name `$self` ist wiederum Konvention, da dieser Name auch in anderen objektorientierten Sprachen für eine Referenz auf

das Objekt selbst verwendet wird. Auf die Referenz, d.h. auf das durch den {}-Operator erzeugte anonyme Hash, wird anschließend die Funktion `bless` angewendet, die dafür sorgt, daß diese Referenz der aktuellen Klasse, also dem aktuellen Package zugeordnet wird. Nur dadurch wird die Referenz zu einem Objekt der entsprechenden Klasse. Die auf diese Weise „gesegnete“ Referenz wird schließlich als Ergebnis zurückgeliefert.

Die Auswirkung der Funktion `bless` kann bis zu einem gewissen Grad beobachtet werden, indem die Referenz `$self` vor und nach dem `bless`-Aufruf ausgegeben wird:

```
package anyClass;
sub new {
   $self={};      # Neues anonymes hash erzeugen
   warn "Vor bless: $self \n";
   bless $self;   # Dieses Hash der Klasse zuordnen
   warn "Nach bless: $self \n";
   return($self); # Referenz als Ergebnis liefern
}
```

Bei einem Aufruf der Methode `new()` erfolgt eine Ausgabe der Art:

```
Vor bless:  HASH(0x7434f4)
Nach bless: anyClass=HASH(0x7434f4)
```

Die Ausgabe zeigt deutlich die Zuordnung der Referenz zu der Klasse nach dem Aufruf von `bless`. Diese Zuordnung ist notwendig, um später über diese Referenz auf Daten und Methoden der Klasse zugreifen zu können.

Neben der Möglichkeit, ein anonymes Hash als Objekt zu definieren, ist es in Perl ebenso möglich, ein anonymes Array zu diesem Zweck einzusetzten. Der Konstruktor aus dem letzten Beispiel sieht jetzt wie folgt aus:

```
package anyClass;
sub new {
   my($self);
   $self=[];      # Neues anonymes Array erzeugen
   bless $self;   # Dieses Array der Klasse zuordnen
   return($self); # Referenz als Ergebnis liefern
}
```

Im Unterschied zum Konstruktor aus dem vorangegangenen Beispiel wird der Variablen `$self` hier eine mit dem `[]`-Operator erzeugte Referenz auf ein anonymes Array zugewiesen, anstelle der Referenz auf ein anonymes Hash, die mittels des {}-Operators erzeugt wird.

Der Unterschied zwischen beiden Varianten besteht darin, wie die Instanz-Variablen des Objekts aussehen, die entweder unter Verwendung des anonymen Hashs oder des anonymen Arrays gebildet werden. Dieser Punkt wird ausführlicher im Abschnitt *Daten* auf Seite 208 behandelt.