Bernhard Steppan

Einstieg in Java 6

Galileo Press

Auf einen Blick

Inhalt

Teil 5 Anhang

21 Werkzeuge 525

3 Objektorientierte Programmierung

»Ich sehe ein Pferd, dann sehe ich noch ein Pferd – dann noch eins. Die Pferde sind nicht ganz gleich, aber es gibt etwas, das allen Pferden gemeinsam ist, und das, was allen Pferden gemeinsam ist, ist die Form des Pferds. Was unterschiedlich oder individuell ist, gehört zum Stoff des Pferds.«
(Jostein Gaarder)

3.1 Einleitung

Mitte der 60er-Jahre des letzten Jahrhunderts kam es zu einer Softwarekrise. Die Anforderungen an Programme stiegen, und die Software wurde dadurch komplexer sowie fehlerhafter. Auf Kongressen diskutierten Experten die Ursachen der Krise und die Gründe für die gestiegene Fehlerrate. Ein Teil der Softwareexperten kam zu dem Schluss, dass die Softwarekrise nicht mit den herkömmlichen Programmiersprachen zu bewältigen sei. Sie begannen deshalb, eine Generation von neuen Programmiersprachen zu entwickeln.

Die Entwickler dieser Sprachen kritisierten an den herkömmlichen Programmiersprachen vor allem, dass sich die natürliche Welt bisher nur unzureichend abbilden lasse. Um dem zu entgehen, gingen sie von natürlichen Begriffen aus, wie sie die Formenlehre der klassischen griechischen Philosophie geprägt hat, und wandelten sie für die Programmierung ab. Da sich alles um den Begriff des Objekts dreht, nannten sie die neue Generation von Sprachen »objektorientiert«.

3.1.1 Grundbegriffe

Alan Kay, einer der Erfinder der Programmiersprache *Smalltalk*, hat die Grundbegriffe der objektorientierten Programmierung folgendermaßen zusammengefasst:

- Alles ist ein Objekt.
- Objekte kommunizieren durch Nachrichtenaustausch.
- Objekte haben ihren eigenen Speicher.
- Jedes Objekt ist ein Exemplar einer Klasse.
- Die Klasse modelliert das gemeinsame Verhalten ihrer Objekte.
- Ein Programm wird ausgeführt, indem dem ersten Objekt die Kontrolle übergeben und der Rest als dessen Nachricht behandelt wird.

3.1.2 Prinzipien

Neben diesen Grundbegriffen sind folgende Prinzipien wichtig:

- Abstraktion
- Vererbung
- Kapselung
- Beziehungen
- Persistenz
- Polymorphie

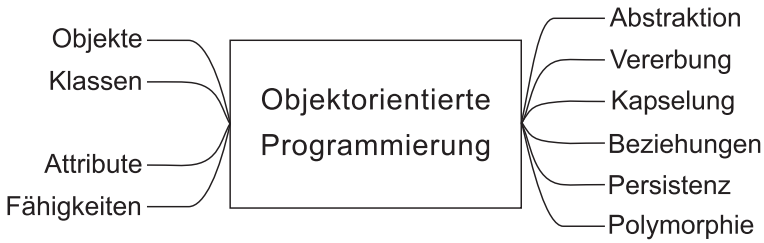

Abbildung 3.1 Hauptbegriffe der objektorientierten Programmierung

3.2 Objekte

Objekte sind für ein Java-Programm das, was Zellen für einen Organismus bedeuten: Aus diesen kleinsten Einheiten setzt sich eine Anwendung zusammen. Objekte haben eine bestimmte Gestalt (Aussehen, Attribute, Kennzeichen) und bestimmte Fähigkeiten (Methoden, Funktionen). Gestalt und Fähigkeiten eines Objekts werden durch seine Erbinformationen bestimmt. Diese Erbinformationen sind der Bauplan, nach dem das Objekt erzeugt wird. In der objektorientierten Programmierung ist der Bauplan eines Objekts seine *Klasse*.

Wenn Sie eine Reihe von Pferden betrachten, fällt Ihnen auf, dass ihnen die prinzipielle *Gestalt* gemeinsam ist. Ihre Unterschiede sind die *Attribute*, die das einzelne Pferd kennzeichnen. Ein Objekt der Klasse *Pferd* ist beispielsweise ein Pferd mit dem Namen *Xanthos*, ein anderes Pferd heißt *Balios*. Beide *Exemplare*[1] haben einen ähnlichen Körperbau (Gestalt) und ähnliche Fähigkeiten. Sie können beispielsweise beide laufen und wiehern.

1 Exemplar und Objekt sind gleichbedeutend. Im Gegensatz dazu ist der Begriff »Instanz« eine Fehlübersetzung (engl. instance: Exemplar) und taucht in diesem Buch deshalb nicht auf.

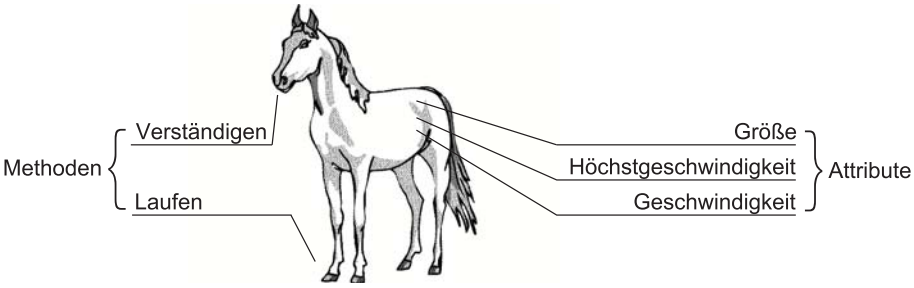

Abbildung 3.2 Jedes Objekt besitzt sein bestimmtes Verhalten und Aussehen.

Die beiden Objekte weisen aber auch einige deutliche Unterschiede auf: *Xanthos* ist weiß und *Balios* braun, und *Xanthos* kann schneller laufen als *Balios*. Obwohl *Xanthos* und *Balios* zur gleichen Klasse gehören, sind nur ihre *prinzipiellen* Fähigkeiten identisch, nicht aber ihre *individuellen* Attribute. Was das bedeutet, wird im nächsten Abschnitt deutlich.

3.3 Klassen

Xanthos und *Balios* gehören zu der Klasse *Pferd*. Die Klasse ist es, die die prinzipielle Gestalt und die Fähigkeiten der beiden Pferde-Objekte festlegt. Man bezeichnet Klassen daher als

- *Bauplan* für Objekte *oder* als
- *Oberbegriff* für verschiedene Objekte (Klassifizierung) *oder* als
- *Schablone* für verschiedene Objekte.

3.3.1 Attribute

Die eingangs erwähnte Klasse *Pferd* soll die Attribute *Größe*, *Höchstgeschwindigkeit* und *Geschwindigkeit* besitzen. Wenn aus dieser Klasse neue Objekte (neue Exemplare) entstehen, besitzen *alle* eine bestimmte *Größe*, eine bestimmte *Höchstgeschwindigkeit* und eine bestimmte *Geschwindigkeit* – aber welche Werte haben diese Attribute? Sie werden erst beim Entstehen (Erzeugen) des Objekts mit den individuellen Werten belegt.

Konstanten

Beispielsweise soll Xanthos über folgende Attribute verfügen: Größe (Stockmaß) 1,90 m, Höchstgeschwindigkeit 65 km/h, Geschwindigkeit 0 km/h. Balios hingegen soll 1,85 m groß sein, maximal 60 km/h schnell laufen können und momentan gerade 5 km/h laufen.

Obwohl beide Pferde nach dem gleichen Bauplan erzeugt worden sind, sind zwei deutlich unterschiedliche Objekte entstanden: Beide sind unterschiedlich groß, können laufen, aber unterschiedlich schnell, beide besitzen eine Geschwindigkeit, aber ein Pferd steht und das andere bewegt sich langsam.

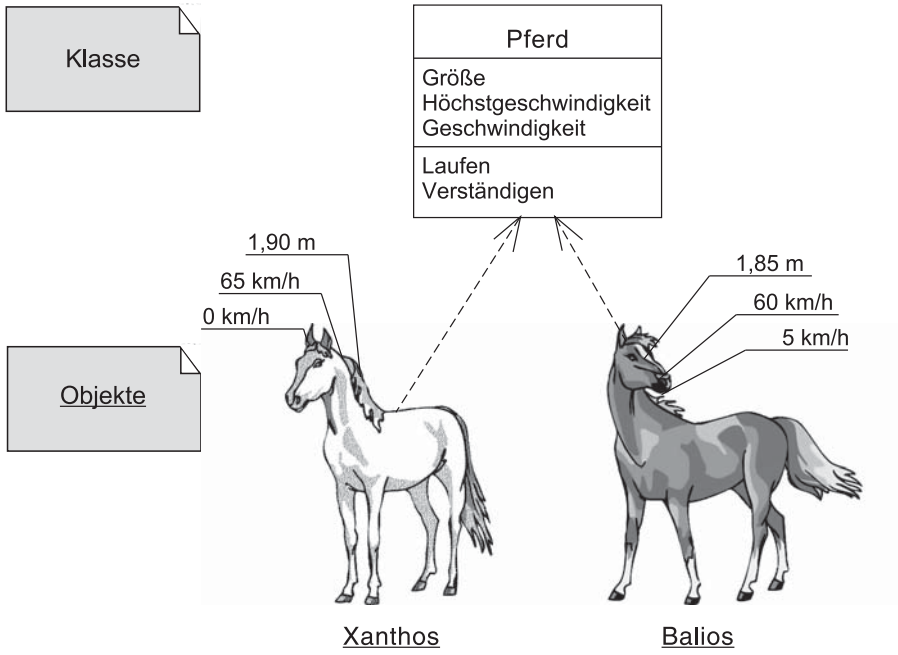

Abbildung 3.3 Die Klasse »Pferd« liefert den Bauplan für Pferde-Objekte.

Zustände

Es ist Ihnen vielleicht aufgefallen, dass bei den bisherigen Attributen der beiden Pferde einige mit festen Werten belegt waren, andere hingegen mit veränderlichen Werten. Die flexiblen Attribute beschreiben den *Zustand* des Objekts. Zum Beispiel beschreibt die *Geschwindigkeit*, wie schnell sich Xanthos gerade bewegt. Der Zustand eines Objekts kann sich im Laufe der Zeit ändern.

Kennungen

Was würde passieren, wenn man Xanthos und Balios so erzeugen würde, dass sie die gleiche *Größe*, die gleiche *Höchstgeschwindigkeit* und die gleiche momentane *Geschwindigkeit* besitzen? Wie könnte man sie dann unterscheiden? In diesem Fall haben beide Objekte zwar individuelle Werte für ihre Attribute bekommen, aber diese sind zufällig gleich. Damit gleichen sich auch die Objekte in einem Programm wie eineiige Zwillinge.

Um die Pferde zu unterscheiden, benötigt man so etwas wie einen genetischen Fingerabdruck. In der Programmierung vergibt der Entwickler eine so genannte Kennung. Diese Kennung ist ein zusätzliches Attribut, bei dem darauf geachtet wird, dass es *eindeutig* ist. Objekte der *gleichen Klasse* besitzen also die gleichen Attribute, aber mit individuellen Werten. Erst die Kennung eines Objekts sorgt dafür, dass das Programm unterschiedliche Exemplare auch dann unterscheiden kann, wenn ihre Attribute zufällig die gleichen Werte besitzen.

3.3.2 Methoden

Angenommen, Sie wollen dem Objekt *Balios* mitteilen, dass es nun springen soll. Im wirklichen Leben geben Sie ihm dazu ein Zeichen. In der objektorientierten Programmierung müssen Sie stattdessen eine Methode des Objekts *Balios* aufrufen. Statt »Methode« werden Sie auch öfter auf die Begriffe »Botschaft« (Smalltalk), »Nachricht« oder »Operation« stoßen, die das Gleiche bedeuten sollen.

Abbildung 3.4 Objekte verständigen sich durch den Austausch von Nachrichten.

Egal, wie der Begriff nun bei den verschiedenen Programmiersprachen und in der Literatur genannt wird, eines ist gleich: Verhaltensweisen wie *Laufen* und *Verständigen* bestimmen die Fähigkeit eines Objekts zu kommunizieren und Aufgaben zu erledigen. Objekte verständigen sich also über Methoden.

Es existiert nicht nur eine Art von Methoden, sondern es gibt folgende fünf Grundtypen: *Konstruktoren* (»Erbauer«), *Destruktoren* (»Zerstörer«), *Mutatoren* (»Veränderer«), *Akzessoren* (»Zugriffsmethoden«) und *Funktionen* (»Tätigkeiten«).

Index