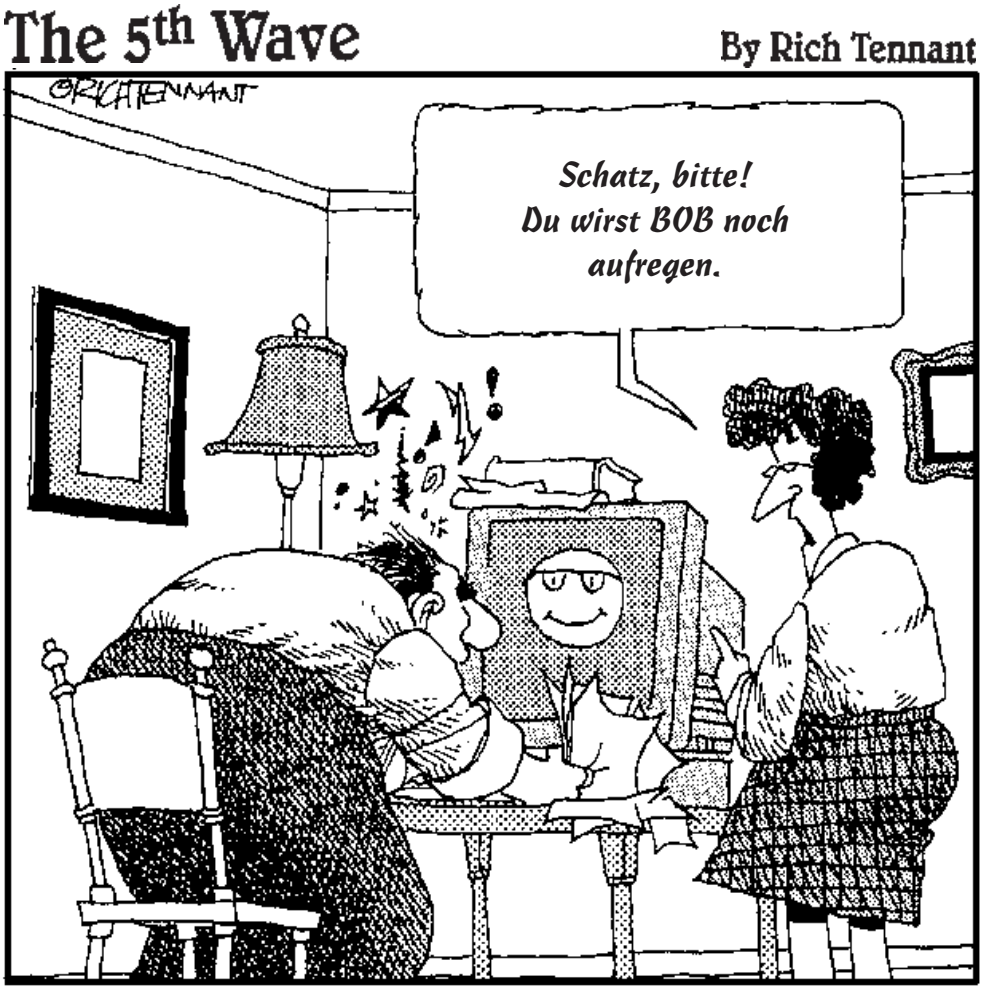
Teil I

Die Grundlagen



In diesem Teil ...

Machen Sie sich mit Java vertraut. Lernen Sie, was Java leisten kann und warum Sie Java verwenden sollten (oder nicht). Wenn Sie nur unklare Vorstellungen davon haben, was Java kann, können Sie sich in diesem Teil Klarheit verschaffen. Wenn Sie nicht wissen, wie Sie ein Java-Programm ausführen können, erfahren Sie in diesem Teil alles, was Sie dazu wissen müssen. Vielleicht haben Sie anderen Leuten erzählt, dass Sie ein Java-Experte sind, und suchen jetzt nach einer Möglichkeit, durch einen Bluff aus dieser selbst gestellten Falle herauszukommen. Falls dies der Fall sein sollte, finden Sie in diesem Teil des Buches einen Schnellkurs in Java (… was nicht heißen soll, dass ich Bluffen gutheiße).

Alles über Java

1

In diesem Kapitel

- Was ist Java?
- Wie wurde Java entwickelt?
- Warum ist Java so attraktiv?
- Wie Sie sich in der objektorientierten Programmierung zurechtfinden können

Ich weiß ja nicht, was Sie von Computern halten, aber für mich sind Computer aus den folgenden beiden einfachen Gründen nützlich:

✔ Wenn Computer arbeiten, fühlen sie keinen Widerwillen, keinen Stress, keine Langeweile und keine Müdigkeit. Computer sind unsere elektronischen Sklaven. Mein Computer ist rund um die Uhr damit beschäftigt, Berechnungen für `SETI@home`, der Suche nach außerirdischer Intelligenz auszuführen. Habe ich Mitleid mit meinem Computer, weil er so hart arbeitet? Beklagt sich der Computer? Geht der Computer vor das Arbeitsgericht? Nein.

  Es ist ganz einfach: Ich gebe Befehle, und der Computer gehorcht. Habe ich dabei Schuldgefühle? Nicht die geringsten. Sollte ich welche haben? Absolut nicht.

✔ Computer bewegen Daten, nicht Papier. Vor nicht allzu langer Zeit wurden Nachrichten noch von berittenen Boten über größere Entfernungen transportiert. Die Nachrichten wurden auf Papier, Pergament, eine Tontafel oder ein anderes physisches Medium geschrieben, das zur jeweiligen Zeit gebräuchlich war.

  Heute betrachten wir diesen Prozess als eine große Verschwendung von Ressourcen, aber das liegt nur daran, dass wir am Anfang des elektronischen Zeitalters stehen. Der Schlüssel liegt darin, dass Nachrichten Daten sind, die nicht an bestimmte physische Medien wie Tinte, Papier und Pferde gebunden sind. Medien sind nur temporäre Träger dieser Daten, doch die Daten selbst sind nicht mit den Datenträgern identisch.

  Das Schöne an Computern ist, dass sie Daten effizient übertragen. Die Datenträger bestehen aus Elektronen und/oder Lichtimpulsen in Kabelnetzen sowie aus elektromagnetischen Wellen in Funknetzen, ohne dass Gegenstände physisch bewegt werden müssten.

Wenn Sie den Datenfluss unter diesem Aspekt betrachten, verschwindet plötzlich der gesamte physische Ballast: Statt Bäume zu fällen, Papier zu produzieren, Gedanken mit Tinte oder Druckerschwärze auf dem Papier festzuhalten, dieses dann zu transportieren usw., werden Gedanken als Daten elektronisch erfasst, gespeichert und übertragen. Der Prozess läuft sehr viel schneller ab, und es wird möglich, viel mehr und viel komplexere Dinge zu tun, als es mit den früheren Verfahren möglich gewesen wäre.

Was leistet Java?

Es wäre schön, wenn diese ganze Komplexität kostenlos wäre; leider ist dies nicht der Fall. Zunächst muss jemand gründlich nachdenken und genau festlegen, was der Computer tun soll. Danach muss jemand die Befehle schreiben, die der Computer ausführen soll.

Beim jetzigen Stand des technischen Wissens ist es nicht möglich, einem Computer Befehle in einer natürlichen Sprache (Deutsch, Englisch usw.) zu geben. In der Science-Fiction gibt es zahlreiche Geschichten von Leuten, die Robotern einfache Befehle geben und damit unerwartete, katastrophale Folgen auslösen. Deutsch und andere natürliche Sprachen eignen sich aus mehreren Gründen nicht zur Kommunikation mit Computern:

- ✔ **Ein deutscher Satz kann mehrdeutig sein.** »Zerkaue eine Woche lang eine Tablette dreimal am Tag.«
- ✔ **Es ist schwierig, einen sehr komplizierten Befehl in Deutsch zu formulieren.** »Verbinden Sie den Flansch A mit der Ausbuchtung B. Achten Sie darauf, nur die äußere Nase von Flansch A in die größere Vertiefung der Ausbuchtung B einzupassen, während Sie die mittlere und die innere Nase in die Vertiefung C einführen.«
- ✔ **Ein deutscher Satz enthält viel Ballast.** »Ein Satz enthält überflüssige Wörter.«
- ✔ **Es ist nicht immer leicht, deutsche Sätze zu verstehen.** »Als Teil dieser Verlagsvereinbarung zwischen John Wiley & Sons, Inc. (»Wiley«) und dem Autor (»Barry Burd«) wird Wiley für die Teillieferung von *Java 2 For Dummies, 2. Auflage* (»das Werk«) die Summe von eintausendzweihundertsiebenundfünfzig Dollar und dreiundsechzig Cents ($1257,63) an den Autor zahlen.«

Um einem Computer mitzuteilen, was er tun soll, müssen Sie eine spezielle Sprache verwenden und in dieser Sprache knappe, eindeutige Anweisungen schreiben. Eine solche Sprache wird als *Programmiersprache* bezeichnet. Die Anweisungen, die in einer solchen Sprache geschrieben werden, werden als *Programm* bezeichnet. Eine Gesamtheit solcher Anweisungen wird auch als *Software* oder *Code* bezeichnet. Beispielsweise könnte der Code für eine Zahlung an den Autor in Java folgendermaßen aussehen:

```
import static java.lang.System.out;

class PayBarry {
    public static void main(String args[]) {

        double checkAmount = 1257.63;
        out.print("Zahlen Sie dem Empfänger ");
        out.print("Dr. Barry Burd ");
        out.print("$");
        out.println(checkAmount);
    }
}
```

Warum sollten Sie Java verwenden?

Zeit zum Feiern! Sie haben zu einem Exemplar von *Java 2 für Dummies* gegriffen und lesen gerade Kapitel 1. Wenn Sie so weitermachen, werden Sie in kürzester Zeit ein Experte in der Java-Programmierung sein; deshalb können Sie in Erwartung Ihres künftigen Erfolgs eine große Party feiern.

Um die Party vorzubereiten, werde ich einen Kuchen backen. Da ich faul bin, werde ich eine fertige Backmischung verwenden. Mal sehen ... rühren Sie die Mischung in Wasser ein, fügen Sie Butter und Eier hinzu ... halt, einen Moment! Ich habe gerade einen Blick auf die Zutatenliste geworfen. Was ist E466? Und was soll ich mit Propylenglykol? Wird das nicht in Kühlschränken verwendet?

Ich werde den Kuchen lieber ganz selber backen. Sicher – das ist etwas schwieriger, aber dann bekomme ich genau das, was ich will.

Computerprogramme funktionieren auf dieselbe Weise. Sie können das Programm eines anderen Programmierers verwenden, oder Sie können das Programm selbst schreiben. Wenn Sie auf Programme eines anderen Programmierers zurückgreifen, können Sie nur das verwenden, was Sie bekommen. Wenn Sie ein eigenes Programm schreiben, können Sie dieses genau auf Ihre Anforderungen zuschneiden.

Das Erstellen von Software ist ein großer, weltweiter Wirtschaftszweig, in dem Firmen, Freiberufler, Hobbyisten und andere Menschen tätig sind. Ein typisches Großunternehmen verfügt über Teams, Abteilungen und Fachbereiche, die Programme für das Unternehmen schreiben. Als Einzelperson können Sie Programme für sich selbst oder für andere Personen schreiben, um Ihren Lebensunterhalt zu verdienen oder weil Sie Spaß daran haben. Täglich werden viele Millionen Zeilen von Code geschrieben. Fast alles, was mit einem Computer getan werden kann, können Sie – die entsprechende Zeit vorausgesetzt – selbst programmieren. (Natürlich kann die »entsprechende Zeit« sehr lang sein, aber darum geht es nicht. Viele interessante und nützliche Programme können in Stunden oder sogar Minuten geschrieben werden.)

Java im Rahmen der historischen Entwicklung

Die folgende Liste gibt einen kurzen Überblick über die Programmierung von Computern:

✔ **1954–1957: FORTRAN wird entwickelt.**

FORTRAN (FORmula TRAnslator) war die erste moderne Programmiersprache. Die Sprache eignet sich speziell zum Schreiben wissenschaftlicher Programme. Jahr für Jahr zählt FORTRAN bei Computerprogrammierern auf der ganzen Welt zu den führenden Programmiersprachen. Der bekannte Computerwissenschaftler Tony Hoare sagte einmal: »Ich weiß nicht, wie die Sprache des Jahres 2000 aussehen wird, aber ich bin mir sicher, dass sie FORTRAN heißen wird.«

- **1959: COBOL wird entwickelt.**

  COBOL (Common Business Oriented Language) wurde speziell für geschäftliche Anwendungen geschaffen. Die Hauptfunktion dieser Sprache besteht darin, größere Mengen gleichartiger Geschäftsdaten (Datensätze, wie beispielsweise Kunden- oder Artikeldaten) effizient zu verarbeiten.

  Innerhalb weniger Jahre entwickelte sich COBOL zur gebräuchlichsten Sprache für die geschäftliche Datenverarbeitung. Selbst heute arbeitet noch ein großer Teil der Software-Branche mit COBOL.

- **1972: Dennis Ritchie entwickelt in den Bell-Laboratorien von AT&T die Programmiersprache C.**

  Das Look-and-Feel der Beispiele in diesem Buch stammt von der Programmiersprache C. C-Code verwendet geschweifte Klammern, `if`-Befehle, `for`-Befehle usw.

  Im Hinblick auf ihre Leistungsstärke ist die Programmiersprache C zum Lösen derselben Probleme geeignet wie FORTRAN, Java oder andere moderne Programmiersprachen. (Man könnte auch in COBOL ein Programm für wissenschaftliche Berechnungen schreiben, aber es wäre etwas umständlich, COBOL zu diesem Zweck zu verwenden.) Der Unterschied zwischen Programmiersprachen besteht nicht darin, wie leistungsstark sie sind, sondern darin, wie gut sie sich zur Lösung bestimmter Probleme eignen und wie leicht sie einzusetzen sind. Gerade die letzte Eigenschaft ist bei Java stark ausgeprägt.

- **1986: Bjarne Stroustrup entwickelt (ebenfalls in den Bell-Laboratorien von AT&T) die Programmiersprache C++.**

  Im Gegensatz zu ihrer Vorgängerin C unterstützt die Sprache C++ das objektorientierte Programmieren. Dies war ein riesiger Fortschritt.

- **23. Mai 1995: Sun Microsystems gibt die erste offizielle Version der Programmiersprache Java frei.**

  Java verbessert die Konzepte von C++. Im Gegensatz zu C++ wurde Java im Hinblick auf den Einsatz im World Wide Web optimiert. Die Java-Philosophie des »Write Once, Run Anywhere« (einmal schreiben, überall ausführen) macht die Sprache zu einem idealen Werkzeug für das Entwickeln von Code, der über das Internet verteilt werden soll.

  Außerdem ist Java eine großartige Allzweck-Programmiersprache. Mit Java können Sie fensterbasierte Anwendungen schreiben, Datenbanken erstellen und nutzen, Handheld-Computer steuern und andere Aufgaben lösen. Innerhalb weniger Jahre haben sich weltweit mehrere Millionen Entwickler für Java entschieden.

- **November 2000: Die Vereinigung amerikanischer Colleges (das *College Board*) kündigt an, dass ab 2003 die Examina für die Vergabe von Studienplätzen für ein Informatikstudium in Java stattfinden werden.**

  Was glauben Sie wohl, welche Sprache die Schüler demnächst in der Schule lernen werden? Java natürlich.

✔ **März 2003: Die Firma SkillMarket (`mshiltonj.com/sm`) berichtet, dass die Nachfrage nach Java-Programmierern die Nachfrage nach C++-Programmierern um 42 Prozent übersteigt.**

Und es gibt noch mehr Nachrichten: Die Nachfrage nach Java-Programmierern übersteigt die kombinierte Nachfrage nach C++- und C#-Programmierern um 10 Prozent, und sie übersteigt die Nachfrage nach Visual Basic-Programmierern um enorme 111 Prozent.

Objektorientierte Programmierung (OOP)

Es ist drei Uhr morgens. Ich träume von dem Geschichtskurs, den ich auf der Highschool vergeigt habe. Der Lehrer schreit mich an: »Du hast noch zwei Tage Zeit, um für die Abschlussprüfung zu lernen, aber das wirst du vergessen. Du wirst es einfach vergessen und dich schuldig, schuldig und nochmals schuldig fühlen.«

Plötzlich klingelt das Telefon. Ich schrecke aus meinem tiefen Schlaf hoch. (Sicher – der Traum über meinen Geschichtskurs war nicht angenehm, aber aufgeweckt zu werden ist noch unangenehmer.) Zunächst lasse ich den Telefonhörer fallen. Nachdem ich ihn endlich wiedergefunden habe, grummele ich: »Hallo, wer da?« Eine Stimme antwortet: »Ich bin Reporter bei *The New York Times*. Ich schreibe einen Artikel über Java und muss alles über diese Programmiersprache wissen. Können Sie mir dies in fünf oder weniger Worten erklären?«

Ich bin zu schläfrig und kann nicht denken. Deshalb sage ich irgendetwas, das mir in den Sinn kommt; dann gehe ich wieder schlafen.

Am nächsten Morgen erinnere ich mich kaum an das Gespräch mit dem Reporter. Tatsächlich weiß ich nicht mehr, wie ich seine Frage beantwortet habe. Habe ich dem Reporter gesagt, was er mit seinem Artikel über Java tun könne?

Ich ziehe meinen Morgenmantel an und gehe zum Briefkasten vor meinem Haus. Als ich die Morgenzeitung in die Hand nehme, fällt mein Blick auf eine riesige Schlagzeile:

Burd nennt Java »eine großartige objektorientierte Sprache «

Objektorientierte Sprachen

Java ist objektorientiert. Was bedeutet das? Im Gegensatz zu Sprachen wie FORTRAN, deren Schwerpunkt darauf liegt, dem Computer Anweisungen der Form »Tue dies/tue das« zu geben, konzentrieren sich objektorientierte Sprachen auf Daten. Natürlich sagen auch objektorientierte Programme dem Computer, was er tun soll. Sie beginnen jedoch damit, die Daten zu organisieren; die Befehle kommen später.

Objektorientierte Sprachen sind besser als »Tue dies/tue das«-Sprachen, weil sie Daten so organisieren, dass sie auf vielfältige Art verwendet werden können. Um die Daten zu verändern, können Sie auf dem aufbauen, was Sie bereits haben, statt Ihre vorherige Arbeit zu verwerfen

und jedes Mal von vorne anzufangen, wenn Sie etwas Neues machen. Obwohl Computerprogrammierer im Allgemeinen recht intelligent sind, brauchten Sie eine Weile, um dieses Verfahren zu entwickeln. Wenn Sie Näheres über die historische Entwicklung wissen wollen, sollten Sie den Einschub *Der verschlungene Weg von FORTRAN bis Java* lesen. (Sie müssen aber kein Schuldgefühl haben, wenn Sie diesen Einschub überspringen.)

Der verschlungene Weg von FORTRAN bis Java

In der Mitte der 50er-Jahre des 20. Jahrhunderts wurde eine Programmiersprache namens FORTRAN entwickelt. FORTRAN war eine gute Sprache, aber sie basierte auf der Idee, dem Computer direkte Anweisungen zu geben: »Computer, tue dies und dann tue das.« (Natürlich waren die Befehle in einem richtigen FORTRAN-Programn viel genauer als »tue dies« oder »tue das«.)

In den folgenden Jahren wurden viele weitere Computersprachen entwickelt, die das »Tue dies/tue das«-Modell von FORTRAN kopierten, unter anderem auch die Programmiersprache C. Natürlich gab es auch Abweichler, die aus diesem »Tue dies/tue das«-Lager ausbrachen. In Sprachen namens SIMULA und SmallTalk verlagerten Programmierer die imperativen »Tue dies«-Befehle in den Hintergrund und konzentrierten sich auf die Beschreibungen der Daten. In diesen Sprachen sagten Sie nicht einfach: »Drucke ein Liste der überzogenen Konten.« Stattdessen sagten Sie zunächst: »Das ist unter einem Konto zu verstehen. Ein Konto hat einen Namen und einen Saldo.« Dann sagten Sie: »So musst du vorgehen, um ein Konto zu fragen, ob es überzogen ist.« Plötzlich standen die Daten im Mittelpunkt. Ein Konto war ein Ding mit einem Namen, einem Saldo und einer Methode, mit der es mitteilen konnte, ob es überzogen war.

Sprachen, die sich zunächst auf die Daten konzentrieren, werden als *objektorientierte* Programmiersprachen bezeichnet. Diese objektorientierten Programmiersprachen sind hervorragende Programmierwerkzeuge. Dies sind die Gründe:

- ✔ Zuerst über die Daten nachzudenken, zeichnet einen guten Computerprogrammierer aus.
- ✔ Sie können die Beschreibungen von Daten erweitern und für verschiedene Zwecke wiederverwenden. Wenn Sie dagegen versuchen, die Arbeitsweise eines alten FORTRAN-Programms zu ändern, zeigen die alten Programme, wie brüchig sie sind. Sie funktionieren nicht mehr.

In den 70er-Jahren wurden objektorientierte Sprachen wie SIMULA und SmallTalk durch Artikel in Fachzeitschriften für Computerhobbyisten populär gemacht. Inzwischen vermehrten sich Sprachen, die auf dem alten FORTRAN-Modell basierten, wie Kaninchen.

Im Jahre 1986 entwickelte Bjarne Stroustrup eine Sprache namens *C++*. Die C++-Sprache verbreitete sich rasch, weil sie eng an die alte C-Sprache angelehnt war und zusätzlich die Möglichkeit der objektorientierten Programmierung bot. Viele Unternehmen gaben

die alte FORTRAN/C-Programmierung auf und machten C++ zu ihrer Standardprogrammiersprache.

Aber C++ hatte einen Mangel. In C++ ist es möglich, alle objektorientierten Funktionen zu missachten und Programme in dem alten FORTRAN/C-Programmierstil zu schreiben. Wenn Sie ein Buchhaltungsprogramm in C++ schreiben wollten, standen Sie vor einer grundlegenden Entscheidung:

- ✔ Sie konnten damit anfangen, »Tue dies«-Befehle für den Computer zu schreiben, und ihn anweisen, eine Liste der überzogenen Konten zu drucken.
- ✔ Sie konnten einen objektorientierten Ansatz wählen und mit der Beschreibung beginnen, was es heißt, ein Konto zu sein.

Einige Leute waren der Auffassung, dass C++ das Beste aus beiden Welten kombinierte. Andere meinten dagegen, dass die erste Welt (die Welt von FORTRAN und C) in einer modernen Programmiersprache nichts zu suchen hätte. Die Wahlmöglichkeit würde Programmierer zu oft dazu verleiten, beim Codieren den falschen Weg zu wählen.

Deshalb schuf James Gosling von Sun Microsystems im Jahre 1995 die Sprache *Java*. Von der Syntax her lehnte er sich an das Look-and-Feel von C++ an, verzichtete aber auf die meisten alten prozeduralen Funktionen von C++. Dann fügte er Funktionen hinzu, die es leichter machten, Objekte zu entwickeln. Alles in allem schuf Gosling eine Sprache, die eine reine und klare objektorientierte Philosophie zum Ausdruck bringt. Wenn Sie in Java programmieren, müssen Sie mit Objekten arbeiten. Und so sollte es auch sein!

Objekte und ihre Klassen

In einer objektorientierten Sprache werden Daten mithilfe von Objekten und Klassen organisiert.

Nehmen wir an, Sie wollten ein Computer-Programm entwickeln, um den Bau der Häuser in einer neuen Siedlung zu verwalten. Die Häuser der Siedlung unterscheiden sich nur geringfügig voneinander. Jedes Haus hat eine bestimmte Außenfarbe, eine Türfarbe, einen bestimmten Einrichtungsstil für die Küche usw. In Ihrem objektorientierten Computer-Programm wird jedes Haus durch ein Objekt repräsentiert.

Objekte sind jedoch nicht alles. Obwohl sich die Häuser geringfügig unterscheiden, werden alle Häuser mit denselben Eigenschaften beschrieben. Beispielsweise verfügt jedes Haus über die Eigenschaft *Außenfarbe* oder die Eigenschaft *Küchenstil*. Deshalb benötigen Sie in Ihrem objektorientierten Programm ein Konstrukt, mit dem Sie alle Eigenschaften zusammenfassen können, die ein Hausobjekt besitzen kann. Dieses Konstrukt, das diese Eigenschaften enthält, wird als *Klasse* bezeichnet.

Meiner Meinung nach ist die Bezeichnung *objektorientiertes Programmieren* unvollständig. Man sollte besser vom *Programmieren mit Klassen und Objekten* sprechen.

Beachten Sie, dass ich das Wort *Klassen* zuerst genannt habe. Warum? Betrachten wir noch einmal das Siedlungsprojekt: Irgendwo auf der Baustelle befindet sich in einem kleinen Büro die Blaupause des Architekten. Eine Blaupause entspricht einer Klasse eines objektorientierten Programms. Eine Blaupause enthält eine Liste der Eigenschaften, die jedes Haus hat. Das Verhältnis zwischen der Blaupause und einem konkreten Haus ist das zwischen einer allgemeinen Eigenschaft und einer speziellen Ausprägung dieser Eigenschaft: Die Blaupause enthält die Eigenschaft *Außenfarbe*, und das konkrete Hausobjekt hat eine graue Außenfarbe. Die Blaupause enthält die Eigenschaft *Küchenstil*, und das konkrete Hausobjekt verfügt über Louis-XIV-Küchenschränke.

Es gibt noch andere Parallelen zwischen den Blaupausen eines Architekten und den Klassen der objektorientierten Programmierung. Die Blaupause des Architekten dient als Grundlage für den Bau vieler Häuser. Die Klassen des Programmierers sind Vorlagen, die zur Konstruktion von Objekten zur Laufzeit des Programms dienen.

Klassen und Objekte stehen also in der folgenden Beziehung zueinander: Der Programmierer definiert eine Klasse, und der Computer benutzt die Klassendefinition, um einzelne Objekte (Exemplare) dieser Klasse zu erstellen.

Welche Vorteile bietet eine objektorientierte Sprache?

Wir wollen das Siedlungsbeispiel des vorangegangenen Abschnitts noch etwas ausbauen: Nehmen wir an, dass Sie bereits ein Computer-Programm geschrieben haben, um die Bauanweisungen für Häuser einer neuen Siedlung zu verwalten. Dann entscheidet der oberste Chef, dass der Bauplan geändert werden soll – die Hälfte der Häuser soll drei, die andere Hälfte vier Schlafzimmer haben.

Ein prozedurales Computer-Programm würde unter anderem die folgenden Anweisungen enthalten:

```
Hebe die Baugrube aus.
Befestige die Seitenwände der Grube mit Beton.
Ziehe die Kellerwände hoch.
...
```

Eine solche Vorgehensweise würde einem Architekten entsprechen, der eine lange Liste von Anweisungen anstelle einer Blaupause erstellt. Um einen solchen Plan zu ändern, müssten Sie die Liste durchsuchen, um die Anweisungen für den Bau der Schlafzimmer zu finden. Die Sache wird noch dadurch erschwert, dass die Anweisungen über die Seiten 234, 324, 287, 394–410, 739, 10 und 2 verstreut sein könnten. Wenn der Baustellenleiter die komplizierten Anweisungen anderer Leute entziffern müsste, wäre die Aufgabe noch schwieriger.

Wenn Sie jedoch mit einer Klasse beginnen, entspricht dies dem Arbeiten mit einer Blaupause. Wenn jemand entscheidet, sowohl Häuser mit drei als auch mit vier Schlafzimmern zu bauen, können Sie mit einer Blaupause namens *Haus-Blaupause* beginnen, die über ein Erdgeschoss und eine erste Etage verfügt, aber auf der ersten Etage keine Innenwände enthält. Dann erstellen Sie zwei weitere Blaupausen für die erste Etage – die eine für das Haus mit drei, die andere für das Haus mit vier Schlafzimmern. (Die neuen Blaupausen erhalten die Namen *Drei-Schlafzimmer-Haus-Blaupause* und *Vier-Schlafzimmer-Haus-Blaupause*.)

Ihre Kollegen aus dem Baugewerbe staunen über Ihr logisches Vorgehen, haben aber noch einige Bedenken: »Wie können Sie eine Blaupause als *Drei-Schlafzimmer-Haus-Blaupause* bezeichnen, wenn die Blaupause nur für die erste Etage und nicht für ein ganzes Haus bestimmt ist?«

Diese Frage haben Sie erwartet und antworten: »Die Blaupause für das Haus mit den drei Schlafzimmern kann einen Vermerk enthalten, der besagt, dass sich die Informationen über die anderen Geschosse in der ursprünglichen *Haus-Blaupause* befinden. Auf diese Weise kann die Blaupause für das Haus mit den drei Schlafzimmern ein ganzes Haus beschreiben. Die Blaupause für das Haus mit den vier Schlafzimmern kann den gleichen Vermerk enthalten. Auf diese Weise ist es möglich, auf die Arbeit zurückzugreifen, die bereits geleistet wurde, um die ursprüngliche Haus-Blaupause zu erstellen, und dadurch sehr viel Geld zu sparen.«

In der Sprache der objektorientierten Programmierung *erben* die Klassen für das Haus mit den drei bzw. den vier Schlafzimmern die Funktionen der ursprünglichen `Haus`-Klasse. Man kann auch sagen, dass die Klassen für das Haus mit den drei bzw. den vier Schlafzimmern die ursprüngliche `Haus`-Klasse *erweitern* (siehe Abbildung 1.1).

Die ursprüngliche `Haus`-Klasse wird als *übergeordnete Klasse* der `Drei-` und `Vier-Schlafzimmer-Haus`-Klassen bezeichnet. Umgekehrt gelten die `Drei-` und `Vier-Schlafzimmer-Haus`-Klassen als *untergeordnete Klassen* der ursprünglichen `Haus`-Klasse. Anders ausgedrückt: Die ursprüngliche `Haus`-Klasse ist die *Oberklasse*, *Parent-Klasse* oder *Elternklasse* der `Drei-` und `Vier-Schlafzimmer-Haus`-Klassen; und umgekehrt sind die `Drei-` und `Vier-Schlafzimmer-Haus`-Klassen die *Unterklassen* oder *Child-Klassen* oder *Kindklassen* der ursprünglichen `Haus`-Klasse (siehe Abbildung 1.1).

Natürlich sind Ihre Kollegen neidisch und wollen mehr über Ihre großartigen Ideen erfahren. Da lassen Sie noch eine Bombe platzen: »Indem wir eine Klasse mit Unterklassen erstellen, können wir die Blaupause auch in Zukunft wiederverwenden. Wenn jemand ein Fünf-Schlafzimmer-Haus haben möchte, können wir unsere ursprüngliche Haus-Blaupause erweitern, indem wir zusätzlich eine Fünf-Schlafzimmer-Haus-Blaupause erstellen. Wir müssen nie wieder Geld für eine Blaupause des ursprünglichen Hauses ausgeben.«

»Aber«, sagt ein Kollege in der hinteren Reihe, »was passiert, wenn jemand im Erdgeschoss einen anderen Grundriss haben will? Werfen wir dann die ursprüngliche Haus-Blaupause weg oder fangen wir an, die ursprüngliche Blaupause zu ändern? Das wird teuer, nicht wahr?«

Selbstsicher antworten Sie: »Wir müssen die ursprüngliche Haus-Blaupause nicht ändern. Wenn jemand in seinem Wohnzimmer einen Whirlpool haben will, können wir eine neue,

kleine Blaupause erstellen, die nur das neue Wohnzimmer beschreibt, und sie als *Whirlpool-im-Wohnzimmer-Haus-Blaupause* bezeichnen. Wenn es um den Rest des Hauses (der nicht zum Wohnzimmer gehört) geht, kann diese neue Blaupause auf die ursprüngliche Haus-Blaupause verweisen.« In der Sprache der objektorientierten Programmierung *erweitert* auch die Whirlpool-im-Wohnzimmer-Haus-Blaupause die ursprüngliche Haus-Blaupause. Auch die Whirlpool-Blaupause ist eine Unterklasse der ursprünglichen Haus-Blaupause. Tatsächlich sind auch die Bezeichnungen *übergeordnete Klasse*, *Elternklasse* und *Kindklasse* anwendbar. Das einzig Neue ist, dass die Whirlpool-Blaupause die Wohnzimmerfunktionen in der ursprünglichen Haus-Blaupause *überschreibt*.

Vor dem Aufkommen der objektorientierten Sprachen befand sich die Software-Entwicklung in einer Krise. Programmierer schrieben Code, entdeckten neue Anforderungen und mussten dann den Code verwerfen und von Grund auf neu anfangen. Dies passierte immer wieder, weil die Programmierer den Code, den sie bereits beschrieben hatten, nicht wiederverwenden konnten. Die objektorientierte Programmierung löste dieses Problem weitgehend.

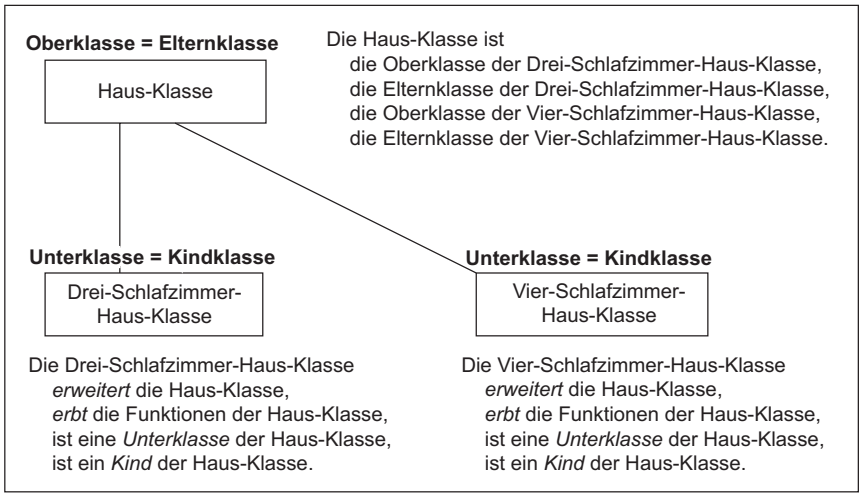


Abbildung 1.1: Terminologie beim objektorientierten Programmieren

Das Verständnis von Klassen und Objekten vertiefen

Wenn Sie in Java programmieren, arbeiten Sie permanent mit Klassen und Objekten. Diese beiden Begriffe sind wirklich wichtig. Deshalb präsentiere ich Ihnen in diesem Kapitel so viele Analogien zu Klassen und Objekten.

Schließen Sie die Augen, und stellen Sie sich für einen Moment einen Stuhl vor ...

Ein Stuhl verfügt über eine Sitzfläche, eine Lehne und Beine. Die Sitzfläche hat eine Form, eine Farbe, eine Polsterung usw. Dabei handelt es sich um Eigenschaften eines Stuhls. Was ich gerade beschrieben habe, macht das Wesen oder den Inbegriff eines Stuhls aus. Übertragen in

die Terminologie der objektorientierten Programmierung bedeutet dies, dass ich die Stuhl-Klasse beschrieben habe.

Wenn Sie sich umschauen, sehen Sie möglicherweise mehrere Stühle. (Falls nicht, stellen Sie sich einfach mehrere Stühle vor.)

Jeder Stuhl ist ein Objekt. Jedes dieser Objekte ist ein Exemplar des ungreifbaren Dings, das wir als *Stuhl-Klasse* bezeichnen. Philosophisch gesprochen repräsentiert die Klasse die Idee der »Stuhlheit« oder das »Wesen des Stuhls«, während jeder einzelne Stuhl ein Objekt ist.

Eine Klasse ist keine Sammlung von Dingen, sondern gewissermaßen der Begriff eines bestimmten Gegenstands. Wenn von der Klasse der Stühle in Ihrem Zimmer die Rede ist, geht es um die Tatsache, dass jeder Stuhl über Beine, eine Sitzfläche, Farben usw. verfügt. Die Stühle mögen sich farblich unterscheiden, aber das spielt keine Rolle. Wenn von einer Klasse von Dingen die Rede ist, geht es um die Eigenschaften, die alle Dinge dieser Klasse besitzen.

Ein Objekt kann als ein Exemplar einer Klasse betrachtet werden. In der Terminologie der objektorientierten Programmierung hat sich dafür auch die Bezeichnung *Instanz* einer Klasse eingebürgert. Wenn Sie ein Java-Programm schreiben, in dem Sie eine Stuhl-Klasse definieren, wird jeder konkrete Stuhl (der Stuhl, auf dem Sie sitzen, der leere Stuhl, der neben Ihnen steht, usw.) als eine *Instanz* der Stuhl-Klasse bezeichnet.

Das folgende Beispiel zeigt eine weitere Variante, über Klassen nachzudenken. Nehmen wir an, dass Sie über drei Bankkonten verfügen (siehe Tabelle 1.1).

Kontonummer	Typ	Saldo
16.1 3154.2 2864.7.	Girokonto	174,87
1011 1234 2122 0000	Kreditkonto	–471,03
16.1 7238.1 3344.7.	Sparkonto	247,38

Tabelle 1.1: Eine Tabelle mit Konten

Betrachten Sie die Zeile mit den Spaltenüberschriften der Tabelle als eine Klasse und jede andere Zeile der Tabelle als ein Objekt. Die Spaltenüberschriften der Tabelle beschreiben die Konto-Klasse.

Danach verfügt jedes Konto über eine Kontonummer, einen Typ und einen Saldo. Oder in der Terminologie der objektorientierten Programmierung: Jedes Objekt in der Konto-Klasse (das heißt, jede Instanz der Konto-Klasse) hat eine Kontonummer, einen Typ und einen Saldo. Beispielsweise zeigt die untere Tabellenzeile ein Objekt mit der Kontonummer *16.1 7238.1 3344.7.*, dem Typ *Sparkonto* und dem Saldo *247,38*. Wenn Sie ein neues Konto eröffnen würden, würden Sie über ein weiteres Objekt verfügen, und die Tabelle würde um eine Zeile wachsen. Das neue Objekt wäre eine Instanz derselben Konto-Klasse.

Wie geht es weiter?

Dieses Kapitel enthält viele allgemeine Beschreibungen von Dingen. Eine allgemeine Beschreibung ist angebracht, wenn Sie gerade anfangen, aber die Dinge noch nicht wirklich verstehen, bis Sie einige spezielle Einzelheiten kennen gelernt haben. Deshalb befassen wir uns in den folgenden Kapiteln zunächst mit den speziellen Einzelheiten.

Bitte blättern Sie deshalb weiter. Das nächste Kapitel wartet schon auf Sie.