IN DIESEM KAPITEL

- Was Java ist
- Woher Java stammt
- Warum Java so cool ist
- Wie Sie sich in der objektorientierten Programmierung zurechtfinden

Kapitel 1
Alles über Java

Sie können von Computern halten, was Sie wollen; soweit es mich betrifft, sprechen zwei gute Gründe für sie:

- ✔ **Wenn Computer ihre Arbeit erledigen, gibt es keinen Widerspruch, sie kennen keinen Stress, empfinden keine Langeweile und ermüden nie.** Ihr Computer kann sieben Tage die Woche 24 Stunden am Tag arbeiten und Berechnungen für `www.climateprediction.net/` durchführen. Oder er kann Zahlen für Rosetta@home knacken – eine Website, die Proteine modelliert, um dazu beizutragen, schwere Krankheiten zu heilen. Tut Ihnen mein Computer leid, weil er so hart arbeiten muss? Beklagt sich der Computer? Nein.

  Sie können Forderungen stellen, dem Computer Anweisungen erteilen und mit der Peitsche knallen. Sollten Sie sich deshalb schuldig fühlen? Nein, natürlich nicht.

- ✔ **Computer bewegen Ideen, kein Papier.** Vor noch nicht allzu langer Zeit mussten Sie einen Boten anheuern, wenn Sie jemandem eine Nachricht zukommen lassen wollten. Dieser Bote stieg auf sein Pferd und lieferte Ihre Nachricht persönlich ab. Die Nachricht stand auf einem Blatt Papier, einem Pergament, einer Tontafel oder auf irgendeinem anderen Medium, das damals gerade verfügbar war.

  Aus heutiger Sicht scheint dieser Vorgang unwirklich zu sein, aber das kommt nur daher, dass Sie und ich im elektronischen Zeitalter bequem vor einem Computer sitzen. Nachrichten sind Ideen und physische Objekte wie Tinte, Papier und Pferde haben nichts oder nur wenig mit echten Ideen zu tun; sie sind nur die Überbringer der Ideen (selbst wenn die Menschen sie jahrhundertelang als Träger von Ideen benutzt haben). Auf jeden Fall gilt, dass Ideen papierlos sind und ohne Pferde und Boten auskommen.

Das Schöne an Computern ist, dass sie Ideen effizient übertragen. Sie befördern nichts als die Ideen, ein paar Photonen und etwas Strom. Und sie erledigen dies ordentlich, ohne zu jammern und ohne zusätzliches Gepäck.

Wenn Sie anfangen, sich ernsthaft mit Ideen zu beschäftigen, geschieht etwas wirklich Angenehmes. Plötzlich verschwindet ein Großteil des Mehraufwands. Statt Papier und Bäume stoßen Sie Zahlen und Konzepte an. Sie können Dinge ohne den Mehraufwand viel schneller erledigen und sind in der Lage, sich mit viel komplexeren Sachen zu beschäftigen denn je.

Was Sie mit Java machen können

Es wäre richtig klasse, wenn diese Vielfalt einfach so vorhanden wäre, aber unglücklicherweise ist das nicht der Fall. Jemand muss schwer nachdenken und entscheiden, was der Computer genau machen soll. Nach dem Nachdenken muss sich jemand hinsetzen und Anweisungen für den Computer schreiben, denen dieser zu folgen hat.

Der heutige Stand der Dinge verbietet es leider, diese Anweisungen in Deutsch oder einer anderen gesprochenen Sprache zu schreiben. Science-Fiction lebt auch von Geschichten, in denen Menschen Robotern einfache Dinge befehlen, die zu desaströsen, unerwarteten Ergebnissen führen. Deutsch und andere Sprachen eignen sich aus mehreren Gründen nicht für eine Kommunikation mit Computern:

- ✔ **Deutsche Sätze können mehrdeutig sein.** »Engländer tragen zu feierlichen Anlässen gerne eine Melone.«
- ✔ **Es ist schwierig, im Deutschen einen komplizierten Befehl in seine logischen Einzelteile zu zerlegen.** »Verbinden Sie Flansch A mit Ausstülpung B und sorgen Sie dafür, dass nur der äußerste Ring von Flansch A mit dem größeren Ende der Ausstülpung B verbunden wird, während der mittlere und der innere Ring des Flansches A an den Dichtungsring C stoßen.«
- ✔ **Ein deutscher Satz schleppt viel Ballast mit sich herum.** »Der Satz enthält überflüssige Wörter.«
- ✔ **Es kann manchmal schwierig sein, Deutsch zu interpretieren.** »Als Teil des Verwertungsvertrages zwischen der Firma Rubbeldiwubbel GmbH & Co. KG (›Rubbel‹) und dem Autor (›Barry Burd‹) zahlt Rubbel den Betrag von eintausendzweihundertfünfundsiebzig Euro und siebzehn Cent (1.275,17 EUR) an Barry Burd als erste Abschlagszahlung für den Auftrag 631-A19-8612, entweder in einer Summe oder in Raten, wie in Absatz 9 Abschnitt 16 dieser Vereinbarung oder ihren nachfolgenden Erweiterungen unter deutschem Recht festgelegt.«

Um einem Computer mitzuteilen, was er machen soll, müssen Sie eine besondere Sprache verwenden und in dieser Sprache knappe, unmissverständliche Anweisungen (auch *Befehle* genannt) schreiben. Eine Sprache dieser Art wird *Programmiersprache* genannt. Eine Folge von Anweisungen, die in solch einer Sprache geschrieben worden ist, wird *Programm*

genannt. Eine andere Bezeichnung für ein Programm ist *Software* oder *Code*. Und so sieht Code aus, wenn er in Java geschrieben worden ist:

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

        double checkAmount = 1257.63;
        System.out.print("Zahlen Sie an ");
        System.out.print("Dr. Barry Burd ");
        System.out.print("€");
        System.out.println(checkAmount);
    }
}
```

Warum Sie Java verwenden sollten

Es gibt einen Grund zum Feiern. Sie haben sich gerade ein Exemplar von *Java für Dummies* besorgt und Sie lesen Kapitel 1. Bei diesem Tempo werden Sie im Nullkommanichts zu einem Experten in der Java-Programmierung. In Fachkreisen hat ein Entwickler normalerweise mehr Verantwortung als ein Programmierer. In diesem Buch verwende ich die Begriffe Programmierer und Entwickler jedoch synonym. Genießen Sie deshalb Ihren voraussichtlichen Erfolg, indem Sie eine große Party veranstalten.

Als Vorbereitung für diese Party will ich einen Kuchen backen. Da ich bequem bin, verwende ich eine fertige Backmischung. Mal sehen … fügen Sie zur Mischung Wasser hinzu, dann Butter und Eier … Halt, einen Moment! Ich habe mir gerade die Liste mit den Zutaten angesehen. Was ist MSG? Und was hat es mit Propylenglykol auf sich? Das wird doch auch als Frostschutzmittel verwendet, oder?

Ich muss meine Pläne ändern und den Kuchen selbst machen. Das ist zwar ein wenig umständlicher, aber so bekomme ich genau das, was ich haben will.

Computerprogramme arbeiten auf die gleiche Weise. Sie können ein Programm verwenden, das von irgendjemandem stammt, oder Sie schreiben ein eigenes Programm. Wenn Sie sich für die erste Variante entscheiden, müssen Sie das nutzen, was Sie bekommen. Wenn Sie Ihr eigenes Programm schreiben, können Sie es auf Ihre Bedürfnisse hin maßschneidern.

Das Schreiben von Computerprogrammen ist zu einer großen, weltweiten Industrie geworden. Firmen machen es, freiberufliche Profis machen es, und es ist zum Hobby vieler geworden; alle möglichen Menschen machen es. Ganz normale große Unternehmen haben Teams, Abteilungen und Bereiche, die Programme für das Unternehmen schreiben. Und auch Sie können Programme schreiben – für sich selbst oder für jemanden anderen, zum Lebensunterhalt oder nur, weil es Spaß macht. Neueste Schätzungen besagen, dass die Zahl der Codezeilen, die auf der ganzen Welt tagtäglich von Entwicklern geschrieben werden, die Zahl der Methan-Moleküle auf dem Planeten Jupiter überschreitet. (Diese Schätzung stammt von mir.) Stellen Sie sich vor, was Sie alles mit einem Computer anstellen können. Und mit dem entsprechenden Zeitaufwand sollten Sie in der Lage sein, dafür Ihr eigenes Programm zu schreiben. (Wobei der »entsprechende Zeitaufwand« relativ groß sein kann, worum es hier aber nicht geht. Viele interessante Programme können in wenigen Stunden oder sogar Minuten geschrieben werden.)

Einen Überblick erhalten: Wie sich Java einordnen lässt

Hier ein kurzer geschichtlicher Verlauf:

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

  FORTRAN war die erste moderne Programmiersprache für Computer. Es entpuppte sich für die wissenschaftliche Programmierung als echter Renner. Jahrelang war FORTRAN weltweit die führende Programmiersprache.

- ✔ **1959: Grace Hopper von Remington Rand entwickelt die Programmiersprache COBOL.**

  Der Buchstabe *B* steht in COBOL für *Business*, und bei COBOL geht es fast ausschließlich um Geschäftliches. Die primäre Aufgabe dieser Sprache ist es, einen Datensatz nach dem anderen, einen Kunden nach dem anderen oder einen Mitarbeiter nach dem anderen zu verarbeiten.

  COBOL wurde innerhalb weniger Jahre die am häufigsten eingesetzte Sprache für die Verarbeitung geschäftlicher Daten.

- ✔ **1972: Dennis Ritchie von AT&T Bell Labs entwickelt die Programmiersprache C.**

  Das Aussehen der Beispielprogramme in diesem Buch hat ihren Ursprung in der Programmiersprache C. Code, der in C geschrieben wird, verwendet geschweifte Klammern, `if`- und `for`-Anweisungen und andere Elemente.

  Wenn es um die Leistungsfähigkeit geht, können Sie C verwenden, um dieselben Probleme zu lösen, die sich mit FORTRAN, Java oder einer anderen modernen Programmiersprache beseitigen lassen. (Sie können auch in COBOL einen wissenschaftlichen Taschenrechner schreiben, aber das wäre ein Akt, den Sie so schnell nicht vergessen werden.) Der Unterschied zwischen den einzelnen Programmiersprachen ist nicht die Leistungsfähigkeit. Der Unterschied liegt darin, wie einfach und bedienerfreundlich eine Sprache ist.

- ✔ **1986: Bjarne Stroustrup (wieder von den AT&T Bell Labs) entwickelt C++.**

  Die Sprache C++ unterstützt, anders als ihr Vorgänger C, objektorientierte Programmierung. Dies stellte einen riesigen Schritt vorwärts dar. (Siehe auch den nächsten Abschnitt in diesem Kapitel.)

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

  Java verbessert die Konzepte von C++. Java unterstützt die objektorientierte Programmierung nicht nur, es *erzwingt die Verwendung* der objektorientierten Programmierung.

Außerdem ist Java eine Programmiersprache, die sich für so gut wie alle Zwecke einsetzen lässt. Ein in Java geschriebenes Programm läuft problemlos auf allen wichtigen Plattformen wie Windows, Macintosh und Linux. Sie können mit Java Anwendungen mit Fenstern schreiben, Datenbanken erstellen und auswerten, Handheld-Geräte kontrollieren und vieles mehr machen. Die Programmiersprache Java erreicht innerhalb von nur fünf kurzen Jahren 2,5 Millionen Entwickler weltweit. (Ich weiß das, denn ich besitze ein entsprechendes T-Shirt, das das beweist.)

- ✔ **November 2000: Java geht zur Schule.**

  In den USA gibt das College Board bekannt, dass ab 2003 alle Computer-Science-Advanced-Placement-Examen auf Java basieren.

- ✔ **2004: Java nimmt den ersten Platz des weltberühmten TIOBE-Index ein und behält diesen für die nächsten 15 Jahre bei.**

- ✔ **Ebenfalls 2004: Java fliegt ins All!**

  Ein Roboterfahrzeug namens *Spirit* führt Java-Code aus, um den Mars zu erkunden.

- ✔ **Januar 2010: Die Oracle Corporation kauft Sun Microsystems auf, was dazu führt, dass Java in die Produkte der Oracle-Familie einzieht.**

- ✔ **August 2017: Oracle kündigt seinen Plan an, alle sechs Monate neue Versionen von Java zu veröffentlichen.**

  Bis dahin wurden neue Java-Versionen nur alle paar Jahre veröffentlicht. Doch auf die Veröffentlichung von Java 9 im September 2017 folgte nun die Einführung von Java 10 im März 2018. Die nächste Version war Java 11 im September 2018.

  Im September 2021 wurde Java 17 als Long-Term-Support-Version (LTS) veröffentlicht. Das bedeutet, dass Oracle verspricht, Java bis mindestens September 2026 reibungslos laufen zu lassen. Diese LTS-Versionen erscheinen alle zwei Jahre, sodass die nächste felsenfeste, kompromisslose Version von Java, also Java 21, im September 2023 erscheint.

  Der neue Release-Zyklus hat der Entwicklung der Programmiersprache Java neuen Schwung verliehen.

- ✔ **Mai 2020: Java feiert seinen 25. Geburtstag.**

  Die Java-Technologie treibt Anwendungen von Unternehmen wie Netflix, Alibaba, Tinder, Uber, PayPal, der New York Times, Pinterest, Slack, Shopify, Twitter, Target und Wells Fargo an.* `Monster.com`, die Website für die Jobsuche, sagt dazu:

  > »Java ist eine der am häufigsten verwendeten Programmiersprachen, daher ist es kaum überraschend, dass sie die Nummer 1 für Tech-Unternehmen ist, die ihren Funktionsumfang benötigen. Laut Oracle läuft Java auf 3 Milliarden Mobiltelefonen, 125 Millionen TV-Geräten und 89 % der Desktop-Computer in den USA. Java ist überall verbreitet, und die Nachfrage nach erfahrenen Entwicklern ist groß.« **

Wir dürfen also beeindruckt sein.

Quellen:

* www.softwaretestinghelp.com/real-world-applications-of-java,
https://newrelic.com/blog/nerd-life/what-you-can-do-with-java,
https://vaadin.com/blog/the-state-of-java,
https://discovery.hgdata.com/product/spring-boot

** www.monster.com/career-advice/article/programming-languages-you-should-know

Objektorientierte Programmierung (OOP)

Es ist drei Uhr morgens. Ich träume von der Geschichtsstunde, die ich in der Highschool verpasst habe. Der Lehrer schreit mich an: »Du hast zwei Tage, um dich auf die Abschlussprüfung vorzubereiten, aber daran wirst du dich nicht erinnern. Du wirst es vergessen und dich schuldig fühlen.«

Plötzlich klingelt das Telefon. Ich werde abrupt aus dem tiefsten Schlaf gerissen. (Klar, ich liebe es nicht unbedingt, vom Geschichtsunterricht zu träumen, aber noch weniger liebe ich es, aus dem Schlaf geholt zu werden.) Als Erstes ließ ich das Telefon auf den Boden fallen. Nachdem ich eine Weile herumgefummelt hatte, um es wiederzufinden, ließ ich ein mürrisches »Hallo, wer ist da?« hören. Eine Stimme antwortete: »Ich bin Reporter bei der Nachrichtenagentur Reuters. Ich schreibe einen Artikel über Java und ich muss alles über diese Programmiersprache wissen – in einem kurzen Satz.«

Ich bin noch zu benebelt. Ich kann nicht denken. Ich sage also das, was mir gerade einfällt, und schlafe dann weiter.

Am nächsten Morgen kann ich mich kaum an die Unterhaltung mit dem Reporter erinnern. Wenn ich ehrlich bin, erinnere ich mich überhaupt nicht mehr daran, wie ich dessen Fragen beantwortet habe. Habe ich irgendwelche seltsamen Dinge gesagt und dann wieder eingeschlafen?

Ich ziehe mich an und laufe nach draußen in die Zufahrt. Während ich die Morgenzeitung aufhebe, fällt mein Blick auf die riesengroße Schlagzeile: **Burd nennt Java »Eine großartige objektorientierte Sprache«.**

Objektorientierte Sprachen

Java ist objektorientiert. Was heißt das? Anders als Sprachen wie FORTRAN, das sich darauf konzentriert, dem Computer befehlende »Mache dies/Mache das«-Anweisungen zu geben, richten objektorientierte Sprachen ihr Hauptaugenmerk auf Daten. Natürlich sagen auch objektorientierte Sprachen einem Computer, was er zu tun hat. Sie beginnen aber mit dem Strukturieren der Daten, und die Befehle kommen dann später.

Objektorientierte Sprachen sind besser als »Mache dies/Mache das«-Sprachen, weil sie Daten auf eine Weise strukturieren, die es dann zulässt, viele Dinge mit diesen Daten

anzustellen. Um die Daten zu modifizieren, können Sie auf das aufbauen, was Sie bereits haben, anstatt erst einmal alles zu verschrotten, was Sie bisher entwickelt haben, und jedes Mal, wenn Sie etwas Neues benötigen, wieder ganz von vorn anzufangen. Obwohl Programmierer in der Regel clevere Menschen sind, dauerte es doch eine Weile, bis sie das herausbekommen haben. Wenn Sie die ganze Geschichte erfahren wollen, schauen Sie sich den Kasten *Der mühsame Weg von FORTRAN zu Java* an.

Der mühsame Weg von FORTRAN zu Java

Mitte der 1950er entwickelte eine Gruppe von Leuten eine Programmiersprache mit dem Namen FORTRAN. Es war eine gute Sprache, aber sie basierte auf der Idee, dass dem Computer direkte, befehlende Anweisungen übermittelt werden. »Computer, mache dies. Und dann, Computer, mache das.« (Natürlich waren die Anweisungen in einem echten FORTRAN-Progamm viel genauer als »Mache dies« oder »Mache das«.)

In den folgenden Jahren entwickelten Teams viele neue Computersprachen, von denen viele das »Mache dies/Mache das«-Modell von FORTRAN kopiert haben. Eine der bekannteren »Mache dies/Mache das«-Sprachen war die Ein-Buchstabe-Sprache C. Natürlich gab es im »Mache dies/Mache das«-Lager auch einige Abtrünnige. Programmierer haben in Sprachen wie SIMULA und Smalltalk die befehlenden »Mache dies/Mache das«-Anweisungen in den Hintergrund verlagert und sich auf Beschreibungen der Daten konzentriert. Bei diesen Sprachen stehen Sie nicht auf und sagen: »Drucke eine Liste aller zweifelhaften Konten.« Stattdessen beginnen Sie, indem Sie sagen: »Dies hier ist das, was Konto bedeutet. Ein Konto hat einen Namen und einen Kontostand.« Und dann sagen Sie: »Und so wird ein Konto gefragt, ob es ein zweifelhaftes Konto ist.« Plötzlich werden die Daten zum König. Ein Konto war ein Ding, das einen Namen, einen Kontostand und einen Weg besaß, über den es Ihnen mitteilen konnte, ob es ein zweifelhaftes Konto war.

Sprachen, deren Hauptaugenmerk auf den Daten liegt, werden objektorientierte Programmiersprachen genannt. Diese objektorientierten Sprachen erzeugen ausgezeichnete Programmierwerkzeuge. Der Grund dafür:

- ✔ Wenn Sie zuerst an die Daten denken, werden Sie zu einem guten Programmierer.
- ✔ Sie können die Beschreibung der Daten erweitern und immer wieder erneut verwenden. Wenn Sie versuchen, alten FORTRAN-Programmen neue Tricks beizubringen, zeigen diese alten Programme, wie empfindlich sie sein können. Sie gehen in die Brüche.

In den 1970ern wurden objektorientierte Sprachen wie SIMULA und Smalltalk in den Computerzeitschriften, die von Hobbyprogrammierern gelesen wurden, beerdigt. In der Zwischenzeit vermehrten sich Sprachen, die auf dem alten FORTRAN-Modell basierten, wie die Kaninchen.

1986 entwickelte ein Typ mit dem Namen Bjarne Stroustrup eine Sprache, die er C++ nannte. Diese Sprache wurde beliebt, weil sie die Terminologie der alten Sprache C mit der besseren objektorientierten Struktur mischte. Viele Firmen wendeten dem alten Stil der FORTRAN/C-Programmierung den Rücken zu und machten C++ zu ihrem Standard.

Aber C++ hatte eine Schwachstelle. Wenn Sie C++ verwendeten, waren Sie in der Lage, alle objektorientierten Funktionen zu umgehen und ein Programm zu schreiben, in dem Sie auf den alten FORTRAN/C-Programmierstil zurückgriffen. Wenn Sie anfingen, ein Buchhaltungsprogramm zu schreiben, konnten Sie sich für einen von zwei Wegen entscheiden:

- ✔ Sie konnten damit anfangen, dass Sie dem Computer direkt »Mache dies/Mache das«-Anweisungen erteilten, indem Sie ihm die mathematische Entsprechung von »Drucke eine Liste aller zweifelhaften Konten und das schnell!« vorlegten.
- ✔ Sie konnten objektorientiert vorgehen und anfangen, indem Sie beschrieben, was es heißt, ein Konto zu sein.

Einige Fachleute waren der Meinung, dass C++ das Beste aus zwei Welten anbietet, während andere argumentierten, dass die erste Welt (FORTRAN und C) nichts in der modernen Programmierung zu suchen hätte. Wenn Sie einem Programmierer die Möglichkeit geben, Code auf eine von zwei Weisen zu schreiben, würde sich diese Person zu oft für den falschen Weg entscheiden.

1995 entwickelte dann James Gosling von Sun Microsystems eine Sprache mit dem Namen *Java*. Als er sich mit Java beschäftigte, griff Gosling auf das Erscheinungsbild von C++ zurück. Aber Gosling nahm die meisten »Mache dies/Mache das«-Funktionen von C++ und warf sie auf den Müll. Dann fügte er Funktionen hinzu, die die Entwicklung von Objekten geschmeidiger machten und erleichterten. Gosling entwickelte alles in allem eine Sprache, deren objektorientierte Philosophie unverfälscht und sauber ist. Wenn Sie in Java programmieren, haben Sie keine andere Wahl, als mit Objekten zu arbeiten. Und so sollte es auch sein.

Objekte und ihre Klassen

In einer objektorientierten Sprache verwenden Sie Objekte *und* Klassen, um Ihre Daten zu ordnen.

Stellen Sie sich vor, dass Sie ein Computerprogramm schreiben, das die (noch im Bau befindlichen) Häuser einer neuen Wohnanlage im Auge behält. Die Häuser unterscheiden sich nur leicht voneinander. Jedes Haus hat einen unverwechselbaren Außenanstrich, einen Innenanstrich, einen Küchenschrank in einem besonderen Stil und so weiter. In Ihrem objektorientierten Programm bildet jedes Haus ein Objekt.

Aber Objekte sind noch nicht alles. Auch wenn sich die Häuser leicht voneinander unterscheiden, so haben sie doch eine Liste mit Merkmalen gemeinsam. So besitzt zum Beispiel jedes Haus ein Merkmal, das man als *Außenanstrich* bezeichnen könnte. Jedes Haus hat ein Merkmal, das *Stil des Küchenschranks* heißen könnte. Sie benötigen in Ihrem objektorientierten Programm ein Stammblatt, das alle Merkmale enthält, die ein Haus besitzen kann. Diese zentrale Liste aller Merkmale wird *Klasse* genannt.

So, nun wissen Sie es. Objektorientierte Programmierung trägt einen falschen Namen. Sie müsste eigentlich *Programmierung mit Klassen und Objekten* heißen.

Beachten Sie, dass ich das Wort *Klassen* an die erste Stelle gesetzt habe. Wie komme ich dazu, mich so etwas zu trauen? Nun, vielleicht bin ich doch nicht verrückt. Erinnern Sie sich noch einmal an die Häuser, die sich noch im Bau befinden. Irgendwo gibt es auf dem Bauplatz in einem Container ein Stammblatt der Merkmale, das auch *Bauplan* genannt wird. Der Bauplan eines Architekten kann mit der Klasse eines objektorientierten Programmierers verglichen werden. Bei einem Bauplan handelt es sich um eine Liste mit Merkmalen, die jedes Haus erhalten soll. Der Bauplan sagt »Außenanstrich«. Das aktuelle Haus-Objekt hat einen grauen Außenanstrich. Der Bauplan sagt »Küchenschrank«. Das aktuelle Haus-Objekt hat einen Küchenschrank im Stil Ludwig XIV.

Die Analogie hört nicht mit einer Liste der Merkmale auf. Zwischen Bauplänen und Klassen gibt es eine weitere wichtige Parallele. Ein Jahr nachdem Sie den Bauplan erstellt haben, verwenden Sie ihn, um zehn weitere Häuser zu bauen. Und so funktioniert das auch mit Klassen und Objekten. Der Programmierer schreibt als Erstes Code, um eine Klasse zu beschreiben. Wenn das Programm dann läuft, erzeugt der Computer aus den Klassen (dem Bauplan) Objekte.

Und damit kennen Sie die wirkliche Beziehung zwischen Klassen und Objekten. Der Programmierer definiert eine Klasse, und der Computer macht aus den Klassen einzelne Objekte.

Was ist das Besondere an einer objektorientierten Sprache?

Stellen Sie sich anhand der Geschichte über den Hausbau aus dem vorherigen Abschnitt vor, dass Sie bereits ein Computerprogramm geschrieben haben, um die Bauanweisungen für neue Häuser zu kontrollieren. Und dann entscheidet der große Boss, den Plan zu ändern – in einen Plan, bei dem die Hälfte der Häuser drei und die andere Hälfte vier Schlafzimmer bekommt.

Wenn Sie im alten FORTRAN/C-Stil programmiert haben, sieht Ihre Anweisung so aus:

```
Graben Sie einen Graben für das Fundament.

Schalen Sie die Seiten des Grabens ein, damit betoniert werden kann.

Bringen Sie an den Seiten Kanthölzer an, die das Gerüst des Fundaments
bilden.
...
```

Hier arbeiten Sie wie ein Architekt, der statt eines Bauplans eine lange Liste mit Anweisungen erstellt. Wenn Sie den Plan ändern wollen, müssen Sie die gesamte Liste abarbeiten, um die Anweisungen zu finden, die mit dem Bau von Schlafzimmern zu tun haben. Und damit das alles dann nicht zu einfach wird, könnten diese Anweisungen über die Seiten xvii, 234, 394–410, 739, 10 und 2 verteilt sein. Und wenn der Bauleiter auch noch die komplizierten Anweisungen einer dritten Person entziffern muss, dauert die Aufgabe zehnmal so lange.

Wenn Sie nun aber mit einer Klasse beginnen, ist das so, als wenn Sie einen Bauplan anfangen. Wenn Sie festlegen, dass es sowohl Häuser mit drei als auch mit vier Schlafzimmern geben soll, beginnen Sie mit einem Bauplan, der HAUS heißt und ein Erdgeschoss und

einen ersten Stock enthält, wobei es in diesem ersten Stock noch keine Innenwände gibt. Dann erstellen Sie zwei Baupläne für den ersten Stock – einen für Häuser mit drei Schlafzimmern und einen für Häuser mit vier Schlafzimmern. Sie nennen diese Baupläne `Drei-Schlafzimmer-Haus` und `Vier-Schlafzimmer-Haus`.

Ihre Kollegen im Bauunternehmen sind über Ihre logische und strukturierte Vorgehensweise angenehm überrascht, aber sie haben Bedenken. Es gibt eine Frage: »Du hast einen der Baupläne `Drei-Schlafzimmer-Haus` genannt. Wieso das, wo es doch nur um ein Stockwerk und nicht um ein ganzes Haus geht?«

Sie lächeln wissend und antworten: »Der Plan mit den drei Schlafzimmern besagt, dass man sich für Informationen über das Erdgeschoss den ursprünglichen Bauplan ansehen soll. Auf diese Weise beschreibt der Drei-Schlafzimmer-Haus-Bauplan ein ganzes Haus. Dasselbe sagt der Vier-Schlafzimmer-Haus-Bauplan aus. Damit sind wir in der Lage, die ganze Arbeit zu nutzen, die wir bereits in den ursprünglichen Bauplan für ein Haus gesteckt haben, und können viel Geld sparen.«

In der Sprache der objektorientierten Programmierung *erben* die Klassen `Drei-` und `Vier-Schlafzimmer-Haus` die Funktionen der ursprünglichen Klasse `Haus`. Sie können auch sagen, dass die Klassen `Drei-` und `Vier-Schlafzimmer-Haus` die ursprüngliche Klasse `Haus` erweitern (siehe Abbildung 1.1).

Die ursprüngliche Klasse `Haus` wird die *Oberklasse* der Klassen `Drei-` und `Vier-Schlafzimmer-Haus` genannt (auch *Superklasse* oder *übergeordnete Klasse* genannt). Umgekehrt sind die Klassen `Drei-` und `Vier-Schlafzimmer-Haus` *Unterklassen* der ursprünglichen Klasse `Haus`. Doch damit nicht genug. Die ursprüngliche Klasse `Haus` wird auch *Elternklasse* der Klassen `Drei-` und `Vier-Schlafzimmer-Haus` genannt. Dementsprechend sind die Klassen `Drei-` und `Vier-Schlafzimmer-Haus` die *Kindklassen* der ursprünglichen Klasse `Haus` (siehe Abbildung 1.1).

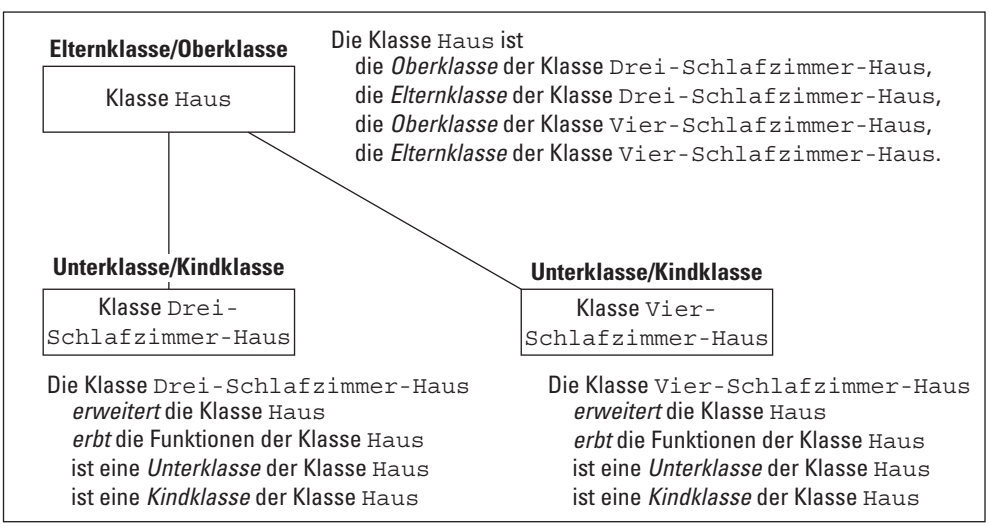

Abbildung 1.1: Die Terminologie der objektorientierten Programmierung

Es muss nicht ausdrücklich darauf hingewiesen werden, dass Ihre Kollegen im Bauunternehmen neidisch sind. Sie werden von vielen Kollegen umringt, die alles über Ihre neue Idee wissen wollen. Das ist genau der richtige Augenblick, um die nächste Bombe hochgehen zu lassen: »Indem wir Klassen mit Unterklassen erstellen, sind wir in der Lage, den Bauplan auch zukünftig wiederzuverwenden. Und wenn dann jemand kommt und ein Haus mit fünf Schlafzimmern haben will, erweitern wir unseren ursprünglichen Bauplan, indem wir einen `Fünf-Schlafzimmer-Haus`-Bauplan anlegen. Wir werden nie wieder Geld für einen originalen Bauplan für ein Haus ausgeben müssen.«

»Aber«, sagt einer der Kollegen in den hinteren Reihen, »was geschieht, wenn jemand ein anderes Erdgeschoss haben möchte? Wirfst du dann den ursprünglichen Bauplan des Hauses auf den Müll oder übermalst du den ursprünglichen Bauplan? Das wird ganz schön teuer, oder?«

Sie antworten in einem vertraulichen Ton: »Wir müssen am ursprünglichen Bauplan nichts ändern. Wenn jemand in seinem Wohnzimmer einen Whirlpool haben möchte, erstellen wir einfach einen neuen kleinen Bauplan, der nur das neue Wohnzimmer beschreibt, und nennen ihn `Whirlpool-im-Wohnzimmer`-HAUS-Bauplan. Dieser neue Bauplan kann auf den ursprünglichen Bauplan HAUS verweisen, um auch weiterhin Zugriff auf die Informationen über den Rest des Hauses (den Teil, der sich außerhalb des Wohnzimmers befindet) zu haben.« In der Sprache der objektorientierten Programmierung *erweitert* der Bauplan `Whirlpool-im-Wohnzimmer-Haus` den ursprünglichen HAUS-Bauplan. Der `Whirlpool`-Bauplan ist außerdem eine Unterklasse des ursprünglichen HAUS-Bauplans. Die gesamte Terminologie wie Oberklasse, Elternklasse und Kindklasse gilt auch hier. Neu ist nur, dass der `Whirlpool`-Bauplan die ursprünglichen Funktionen des Wohnzimmers *überschreibt*.

In der Zeit vor den objektorientierten Sprachen durchlebte die Programmierwelt eine Krise der Softwareentwicklung. Programmierer schrieben Code, und dann entdeckten sie neue Bedürfnisse und mussten ihren Code auf den Müll werfen und ganz von vorn anfangen. Dies geschah immer wieder, weil der Code, den die Programmierer geschrieben haben, nicht wiederverwendet werden konnte. Dies alles wurde durch die objektorientierte Programmierung zum Besseren geändert (und wie Burd gesagt hat, ist Java »eine großartige objektorientierte Sprache«).

Objekte und Klassen sind einfach überall

Wenn Sie in Java programmieren, arbeiten Sie ständig mit Klassen und Objekten. Diese beiden Konzepte sind lebenswichtig. Das ist auch der Grund dafür, warum ich Ihnen in diesem Kapitel eine Analogie von Klassen und Objekten nach der anderen um die Ohren haue.

Schließen Sie für eine Minute Ihre Augen, und denken Sie darüber nach, was es für einen Gegenstand bedeutet, ein Stuhl zu sein.

Ein Stuhl hat eine Sitzfläche, eine Rückenlehne und Beine. Jede Sitzfläche hat eine Form, eine Farbe, einen Härtegrad und weitere Eigenschaften. Dies sind die Eigenschaften eines Stuhls. In der objektorientierten Terminologie beschreibe ich die Klasse `Stuhl`.

Werfen Sie nun einen Blick über den Rand dieses Buches hinaus, und schauen Sie sich einmal im Zimmer um. (Wenn Sie gerade nicht in einem Zimmer sitzen, stellen Sie es sich vor.)

Im Zimmer gibt es mehrere Stühle, und jeder Stuhl ist ein Objekt. Jedes dieser Objekte ist ein Beispiel dieses ätherischen Elements, das als `Stuhl`-Klasse bezeichnet wird. Und genau so funktioniert das – die Klasse ist die Vorstellung von Bestuhlung, und jeder einzelne Stuhl ist ein Objekt.

Eine Klasse ist nicht wirklich eine Sammlung von Elementen. Stattdessen handelt es sich bei einer Klasse um die Idee hinter einer bestimmten Art von Elementen. Wenn ich über die Klasse Stühle in Ihrem Zimmer spreche, meine ich die Tatsache, dass jeder Stuhl Beine, eine Sitzfläche, eine Farbe und anderes hat. Die verschiedenen Stühle im Raum können eine unterschiedliche Farbe haben, aber das macht nichts. Wenn Sie über eine Klasse von Elementen sprechen, geht es um die Eigenschaften, die jedes dieser Elemente besitzt.

Es macht Sinn, sich ein Objekt so vorzustellen, als ob es die greifbare Instanz einer Klasse sei. Und es ist wirklich so, dass die offizielle Terminologie mit dieser Denkweise übereinstimmt. Wenn Sie ein Java-Programm schreiben, in dem Sie eine Klasse `Stuhl` definieren, wird jeder reale Stuhl (der Stuhl, auf dem Sie gerade sitzen, der leere Stuhl direkt neben Ihnen und so weiter) als *Instanz* der Klasse `Stuhl` bezeichnet.

Hier kommt ein weiterer Weg, um sich ein Bild von einer Klasse zu machen. Stellen Sie sich eine Tabelle vor, die Ihre drei Bankkonten wiedergibt (siehe Tabelle 1.1).

Kontonummer	Kontoart	Kontostand
1613154228647	Giro	174,87
1011123421220000	Kredit	–471,03
1617238133447	Sparen	247,38

Tabelle 1.1: Eine Tabelle mit Bankkonten

Stellen Sie sich die Spaltenüberschriften als eine Klasse vor, und denken Sie an die einzelnen Zeilen der Tabelle als Objekte. Die Spaltenüberschriften der Tabelle beschreiben die Klasse `Konto`.

Jedes Konto verfügt entsprechend der Spaltenüberschriften der Tabelle über eine Kontonummer, eine Kontoart und einen Kontostand. Wenn wir das in der Terminologie der objektorientierten Programmierung neu formulieren, hat jedes Objekt in der Klasse `Konto` (das heißt, jede Instanz der Klasse `Konto`) eine Kontonummer, eine Kontoart und einen Kontostand. Daraus folgt, dass es sich bei der untersten Zeile der Tabelle um ein Objekt mit der Kontonummer *1617238133447* handelt. Dasselbe Objekt besitzt die Kontoart *Sparen* und einen Kontostand von *247,38*. Wenn Sie ein neues Konto eröffnen würden, hätten Sie ein weiteres Objekt, und die Tabelle würde um eine Zeile wachsen. Das neue Objekt wäre eine Instanz derselben Klasse `Konto`.