IN DIESEM KAPITEL

- Worum es bei der Programmierung von Computern eigentlich geht
- Software zum Schreiben von Programmen
- Integrierte Entwicklungsumgebung an den Start bringen

Kapitel 1
Erste Schritte

Computerprogrammierung? Was soll das sein? Ist das was Technisches? Tut es weh? Ist das politisch korrekt? Hat Google da seine Finger mit drin? Warum sollte man so etwas machen? Und warum gerade ich? Kann man das lernen?

Worum es eigentlich geht

Sie haben Ihren Computer wahrscheinlich bereits zur Textverarbeitung benutzt: einen Brief geschrieben und ausgedruckt und diesen Ausdruck dann an jemanden geschickt, der Ihnen nahesteht. Und wenn Sie einfachen Zugang zu einem Computer haben, dann haben Sie wahrscheinlich auch schon im Internet gesurft: Sie haben eine Webseite aufgerufen und dort auf einen Link geklickt und sind so auf eine andere Seite gelangt. War ja einfach, oder?

Einfach war das im Grunde genommen in erster Linie, weil jemand dem Computer beigebracht hat, was er tun soll. Wenn Sie hingegen einen fabrikneuen Computer nehmen und ihm nicht mitteilen, was er zu tun hat, dann kann er keine Texte verarbeiten, keine Internetseiten aufrufen, kurzum: Er kann gar nichts. Denn ein Computer kann nur Anweisungen befolgen, die ihm gegeben werden.

Stellen Sie sich nun vor, Sie schreiben in Microsoft Word einen großen Roman und gelangen ans Ende einer Zeile (und nein, damit meine ich nicht das Satzende). Wenn Sie nun das nächste Wort schreiben, springt der Cursor automatisch in die nächste Bildschirmzeile. Wie funktioniert das eigentlich?

Da hat jemand ein *Computerprogramm* geschrieben: eine Abfolge von Anweisungen, denen der Computer entnimmt, was er tun soll. Eine andere Bezeichnung für ein solches Programm (oder einen Teil eines Programms) ist *Code*. Listing 1.1 zeigt, wie ein Auszug aus dem Code von Microsoft Word aussehen könnte.

```
if (spaltennummer > 60) {
    umbrechen();
} else {
    weiterInZeile();
}
```

Listing 1.1: Codezeilen in einem Computerprogramm

Wenn Sie Listing 1.1 in richtiges Deutsch übersetzen, steht dort ungefähr Folgendes:

```
Wenn die Spaltennummer größer als 60 ist,
    dann Umbruch durchführen.
Sonst (also wenn die Spaltennummer nicht größer als 60 ist)
    in derselben Zeile weiterschreiben.
```

Es muss jemanden geben, der Code in der Art des in Listing 1.1 gezeigten schreiben muss. Dieser Code bildet gemeinsam mit mehreren weiteren Millionen Codezeilen ein Programm, das Microsoft Word heißt.

Und wie sieht es beim Surfen im Internet aus? Sie klicken dort auf einen Link, der Sie direkt zu `Facebook.com` bringen soll. Hinter den Kulissen hat jemand hierfür den folgenden Code geschrieben:

```
Zu <a href="https://www.facebook.com">Facebook</a> wechseln.
```

Es gibt also Menschen, die solche Programme schreiben. Und diese Leute heißen eben *Programmierer*.

Einen Computer mit Anweisungen füttern

Bei allem, was Sie an einem Computer tun, kommt Code zum Einsatz. Viel Code. So ist beispielsweise jedes Computerspiel nichts anderes als eine große (lies: unfassbar große) Menge Computercode. Und irgendjemand hat irgendwann einmal das Spielprogramm geschrieben:

```
if (person.beruehrt(goldring)) {
    person.erhaeltPunkte(10);
}
```

Zweifelsohne verfügen Menschen, die Programme schreiben, über wertvolle Fähigkeiten. Es handelt sich dabei im Wesentlichen um zwei besondere Merkmale:

- ✔ Diese Menschen wissen genau, wie sie komplexe Probleme in kleinere, schrittweise zu bearbeitende Prozeduren unterteilen.
- ✔ Sie verstehen, diese Schritte in einer sehr präzisen Sprache auszudrücken.

Eine Sprache, in der solche Schritte beschrieben werden können, heißt *Programmiersprache*, und Java ist nur eine von mehreren Tausend nützlichen Programmiersprachen. Was Sie oben in Listing 1.1 gesehen haben, ist in der Programmiersprache Java abgefasst.

Sie haben die Wahl

In diesem Buch geht es zwar nicht um die Unterschiede zwischen Programmiersprachen, aber Sie sollen auch Code in anderen Sprachen kennenlernen, um einen Eindruck von den Grundprinzipien zu erhalten. So gibt es beispielsweise eine Sprache namens Visual Basic, deren Code sich ein wenig von in Java geschriebenem Code unterscheidet. Ein Auszug aus einem Visual-Basic-Programm könnte etwa so aussehen:

```
If spaltennummer > 60 Then
    Call umbrechen
Else
    Call weiterInZeile
End If
```

Code in Visual Basic sieht verglichen mit dem Java-Code aus Listing 1.1 schon fast wie normales Englisch aus. Und jetzt sehen Sie sich unter dieser Prämisse einmal denselben Code in der Programmiersprache COBOL an:

```
IF SPALTENNUMMER IS GREATER THAN 60 THEN
    PERFORM UMBRECHEN
ELSE
    PERFORM WEITER-IN-ZEILE
END-IF.
```

Am anderen Ende des Spektrums befinden sich Sprachen wie Forth. Nachfolgend gezeigt ist ein kurzes Forth-Programm mit Ein- und Ausgabe:

```
: WRAP?  60  > IF  UMBRECHEN?  ELSE WEITER_IN_ZEILE? THEN ;
```

Computersprachen können sich beträchtlich voneinander unterscheiden, aber die Grundprinzipien sind doch weitgehend identisch. Wenn Sie mit `IF SPALTENNUMMER IS GREATER THAN 60` klarkommen, werden Sie sich auch an `if (spaltennummer > 60)` relativ schnell gewöhnen. Sie müssen im Grunde genommen im Kopf lediglich einen Symbolsatz durch einen anderen ersetzen. Am Ende schreiben Sie dann Dinge wie `if (spaltennummer > 60)` im Schlaf.

Wie Anweisungen aus Ihrem Kopf in den Prozessor gelangen

Beim Entwickeln eines neuen Computerprogramms durchlaufen Sie einen mehrstufigen Prozess. Bei diesem Prozess kommen drei wichtige Tools zum Einsatz:

- ✔ **Compiler.** Ein Compiler übersetzt Ihren Code in vom Computer lesbare (und damit von Ihnen nicht mehr unbedingt lesbare) Anweisungen.

- ✔ **Virtuelle Maschinen.** Eine virtuelle Maschine verarbeitet die computerlesbaren Anweisungen.
- ✔ **API.** Eine API (Application Programming Interface, Programmierschnittstelle) enthält bereits vorab geschriebenen Code, der sich als sehr nützlich erweisen kann.

In den nächsten Abschnitten sollen diese drei Tools beschrieben werden.

Code übersetzen

Sie haben vielleicht schon davon gehört, dass Computer ausschließlich mit Nullen und Einsen rechnen. Das mag gewiss stimmen, aber was heißt das eigentlich? Es bedeutet, dass Schaltkreise in Computern mit Buchstaben erst einmal gar nichts anfangen können. Wenn das Wort *Start* auf Ihrem Bildschirm angezeigt wird, wird es vom Computer selbst als Zeichenfolge `01010011 01110100 01100001 01110010 01110100` gespeichert. Dass Sie hier ein motivierendes Wort mit fünf Buchstaben sehen, ist lediglich Ihrer Interpretation der Bildpunktformen auf dem Bildschirm zu verdanken – mehr nicht. Computer unterteilen alles in sehr grundlegende und schlecht zu lesende Abfolgen von Nullen und Einsen und setzen am Ende alles wieder so zusammen, dass der Mensch mit den Ergebnissen umgehen kann.

Was also geschieht nun, wenn Sie ein Computerprogramm schreiben? Ganz klar: Dieses Programm muss in Nullen und Einsen übersetzt werden. Der offizielle Begriff für diesen Übersetzungsvorgang lautet *Kompilierung*. Ohne Kompilierung kann der Computer Ihr Programm nicht ausführen.

Ich habe den Code aus Listing 1.1 spaßeshalber einmal kompiliert und dann mit einem Trick dafür gesorgt, dass ich die resultierenden Nullen und Einsen anzeigen kann. Das Ergebnis sehen Sie in Abbildung 1.1.

```
11001010 11111110 10111010 10111110 00000000 00000000
00000000 00101110 00000000 00010101 00001010 00000000
00000101 00000000 00010000 00001010 00000000 00000100
00000000 00010001 00001010 00000000 00000100 00000000
00010010 00000111 00000000 00010011 00000111 00000000
00010100 00000001 00000000 00000110 00111100 01101001
01101110 01101001 01110100 00111110 00000001 00000000
00000011 00101000 00101001 01010110 00000001 00000000
00000100 01000011 01101111 01100100 01100101 00000001
00000000 00001111 01001100 01101001 01101110 01100101
01001110 01110101 01101101 01100010 01100101 01110010
01010100 01100001 01100010 01101100 01100101 00000001
00000000 00001011 01100100 01101001 01110011 01110000
01101100 01100001 01111001 01010111 01101111 01110010
01100100 00000001 00000000 00000100 00101000 01001001
00101001 01010110 00000001 00000000 00001110 01110111
01110010 01100001 01110000 01010100 01101111 01001110
01100101 01111000 01110100 01001100 01101001 01101110
01100101 00000001 00000000 00010000 01100011 01101111
01101110 01100100 01101001 01101110 01110101 01100101
01010011 01100001 01101101 01100101 01001100 01101001
01101110 01100101 00000001 00000000 00001010 01010011
01101111 01110101 01110010 01100011 01100101 01000110
```

Abbildung 1.1: Mein Computer versteht Nullen und Einsen. Ich nicht.

Für den kompilierten Kuddelmuddel in Abbildung 1.1 gibt es viele unterschiedliche Bezeichnungen:

- ✔ Die meisten Java-Programmierer nennen so etwas *Bytecode*.
- ✔ Ich spreche häufig von *.class-Dateien*, denn in Java wird Bytecode in Dateien gespeichert, deren Namen `DiesUndJenes.class` lauten.
- ✔ Zur Unterscheidung der beiden Codeformen sprechen Java-Programmierer bei so etwas wie Listing 1.1 vom *Quellcode*, während die Übersetzung in Abbildung 1.1 *Objektcode* heißt.

Abbildung 1.2 veranschaulicht die Beziehung zwischen Quell- und Objektcode. Zuerst schreiben Sie Quellcode, und dann bringen Sie Ihren Computer dazu, aus diesem Quellcode Objektcode zu erstellen. Zu diesem Zweck verwendet der Computer ein spezielles Softwaretool: den *Compiler*.

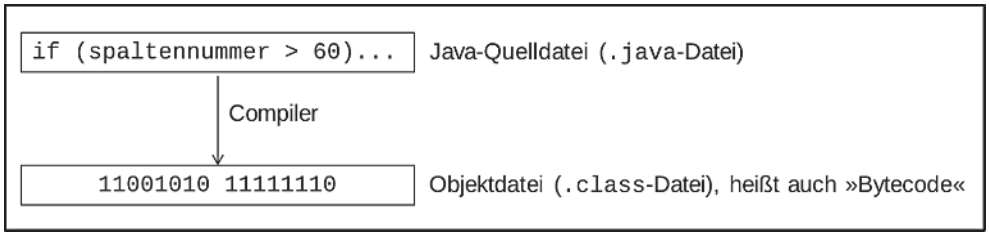


Abbildung 1.2: Der Computer kompiliert Quellcode, um Objektcode zu erstellen.

Auf der Festplatte Ihres Computers finden Sie unter Umständen eine Datei, die `javac` oder `javac.exe` heißt. Dies ist die Datei, die den Compiler enthält. (Und wissen Sie was? `javac` steht für »Java-Compiler«!) Als Java-Programmierer werden Sie Ihren Computer häufig anweisen, neuen Objektcode zu erstellen. Diesen Wunsch erfüllt Ihnen Ihr Computer gerne, indem er hinter den Kulissen die Anweisungen in der Datei `javac` ausführt.

Code ausführen

Vor ein paar Jahren war ich mal eine Woche lang in Kopenhagen. Ich besuchte eine Freundin, die Dänisch wie Englisch fließend spricht. Während wir uns in einem Park miteinander unterhielten, nahm ich nebenbei wahr, dass ein paar Kinder uns interessiert beäugten. Ich spreche nicht ein Wort Dänisch, weswegen ich annahm, dass diese Kinder sich über irgendein für die dänische Jugend alltägliches Thema austauschten.

Darauf meinte meine Freundin unvermittelt, dass die Kinder sich gar nicht auf Dänisch unterhielten. »In welcher Sprache denn?«, erwiderte ich.

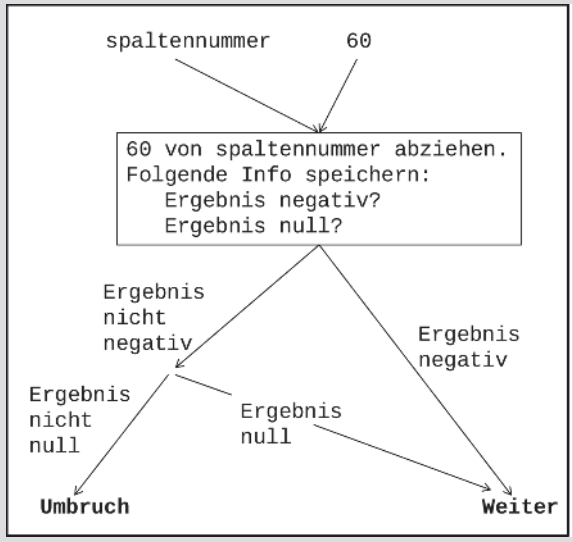
»Sie reden Unsinn«, sagte sie. »Es sind einfach irgendwelche Silben. Sie verstehen kein Englisch, und deswegen ahmen sie deine Sprache einfach nach.«

Zurück zur Sache. Wenn ich mir Abbildung 1.1 so ansehe, bin ich versucht, mich über die Sprache meines Computers lustig zu machen. Dann würde ich genau das Gleiche tun wie

Und was ist jetzt Bytecode?

Sehen Sie sich Listing 1.1 und dessen Übersetzung in Bytecode in Abbildung 1.1 an. Nun könnten Sie vielleicht glauben, dass eine Bytecodedatei nichts anderes als ein Kryptogramm ist – also eine Version des Quellcodes, bei der die Nullen und Einsen die einzelnen Buchstaben in Wörtern wie `if` oder `else` ersetzen. Diese Annahme ist vollkommen falsch. Vielmehr ist der wichtigste Aspekt einer Bytecodedatei die Kodierung der Programmlogik.

Die Nullen und Einsen in Abbildung 1.1 beschreiben den Datenfluss von einem Teil Ihres Computers in einen anderen. Die folgende Abbildung soll dies veranschaulichen. Denken Sie aber daran, dass es sich lediglich um eine konzeptionelle Darstellung handelt: Ihr Computer orientiert sich nicht an dieser (oder irgendeiner anderen) Abbildung, sondern entscheidet auf Basis gelesener Nullen und Einsen, was als Nächstes zu tun ist.



Insofern lohnt es sich auch nicht, die Details meines Versuchs, die Abläufe grafisch darzustellen, auswendig zu lernen. Was Sie meiner Anordnung von Text, Kästchen und Pfeilen hingegen entnehmen sollten, ist, dass der Bytecode (also der Inhalt einer `.class`-Datei) eine vollständige Beschreibung der Operationen enthält, die der Computer durchführen soll. Wenn Sie ein Computerprogramm schreiben, beschreibt der Quellcode eine Gesamtstrategie – sozusagen das große Ganze. Der kompilierte Bytecode wandelt diese Gesamtstrategie in Hunderte winziger Details um – jeweils eines pro Schritt. Wenn das Programm auf Ihrem Computer »läuft«, untersucht der Computer diesen Bytecode und führt jeden einzelnen dieser Schritte aus.

die Jungen und Mädchen in Kopenhagen. Während der Bytecode für mich ohne Aussage ist, kann mein Computer damit durchaus etwas Sinnvolles anfangen. Wenn die in Abbildung 1.1 gezeigten Nullen und Einsen die Schaltkreise meines Computers durchziehen, »denkt« der Computer so wie in Abbildung 1.3 gezeigt.

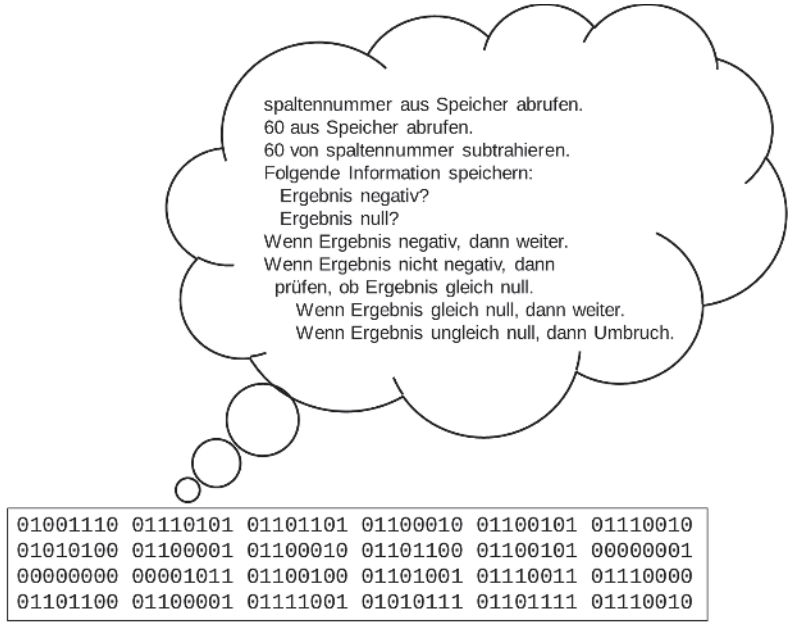


Abbildung 1.3: Was ein Computer aus einer Bytecodedatei macht

Während, wie wir wissen, Computer nicht im eigentlichen Sinne »denken«, können Sie durchaus die in Abbildung 1.3 gezeigten Anweisungen ausführen. Bei vielen Programmiersprachen wie etwa C++ oder COBOL macht der Computer genau das, was ich ihm sage. Er nimmt sich den Objektcode vor und führt exakt aus, was darin beschrieben ist.

So funktionieren viele Programmiersprachen – Java allerdings nicht. Bei Java muss der Computer vielmehr eine andere Abfolge von Anweisungen ausführen. Das sieht dann so aus wie in Abbildung 1.4 gezeigt.

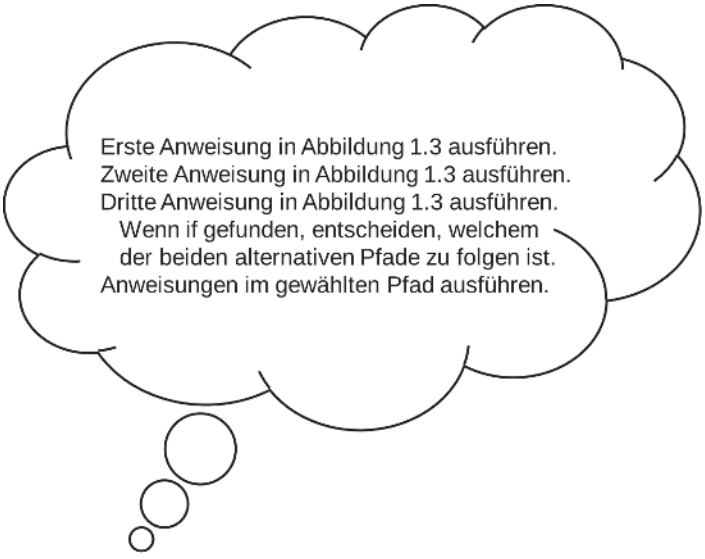


Abbildung 1.4: So führt ein Computer ein Java-Programm aus.

Anhand der Anweisungen in Abbildung 1.4 weiß der Computer, wie er anderen Anweisungen folgen soll. Statt mit `Get columnNumber from memory` zu beginnen, lautet die erste Anweisung des Computers: »Tue, wozu du in der Bytecodedatei angewiesen wirst.« (Wobei dann in der Bytecodedatei die erste Anweisung natürlich gleich wieder `Get columnNumber from memory` lautet.)

Es gibt auch für die Ausführung der in Abbildung 1.4 gezeigten Anweisungen eine spezielle Software. Diese heißt *Java Virtual Machine* oder kurz JVM. Die JVM geleitet Ihren Computer durch die Ausführung bestimmter Bytecodeanweisungen. Wenn Sie ein Java-Programm ausführen, startet Ihr Computer eigentlich die JVM. Diese untersucht dann Ziffer für Ziffer Ihren Bytecode und führt die dort beschriebenen Anweisungen aus.

Es gibt zur Beschreibung der JVM eine Menge guter Metaphern. Sie können sie sich als Stellvertreter, als Laufburschen oder als Verbindungsmann vorstellen. Wie auch immer: Am Ende steht eine Situation, wie sie in Abbildung 1.5 gezeigt ist. Die linke Seite (a) stellt das Prinzip der meisten Programmiersprachen dar: Der Computer führt den Objektcode aus. Rechts (b) sehen wir, wie sich die Angelegenheit bei Java gestaltet: Der Computer führt die JVM aus und diese verarbeitet die Anweisungen im Bytecode.

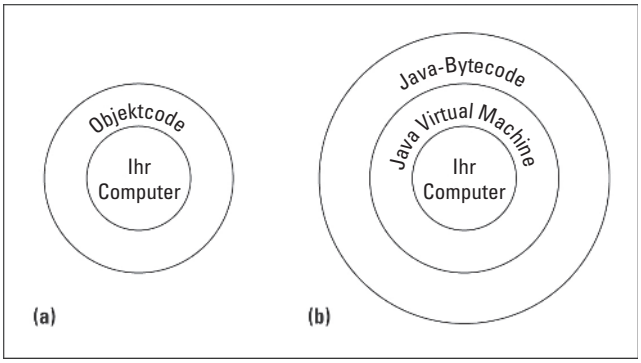


Abbildung 1.5: Verfahren zur Ausführung von Computerprogrammen

Auf der Festplatte Ihres Computers finden Sie möglicherweise Dateien, die `javac` und `java` (beziehungsweise `javac.exe` und `java.exe`) heißen. Die Datei `java` (beziehungsweise `java.exe`) enthält die weiter oben in Abbildung 1.4 gezeigten Anweisungen, nämlich die für die JVM. Als Java-Programmierer werden Sie auf Ihrem Computer häufig Java-Programme ausführen. Zu diesem Zweck startet Ihr Computer dann die Datei `java` und führt darin die Programmanweisungen aus.

Gebrauchsfertiger Code

In den frühen 1980er-Jahren arbeitete mein Schwippschwager Chris für ein Computersoftwareunternehmen. Dort wurde Code für Textverarbeitungssysteme entwickelt. (Damals kaufte man, wenn man Dokumente ohne Schreibmaschine abfassen wollte, einen »Computer«, der nichts anderes beherrschte als Textverarbeitung.) Nun beschwerte sich Chris darüber, dass er denselben alten Code wieder und wieder neu schreiben sollte. »Erst schreibe

»Write Once, Run Anywhere«

Als Java 1995 vorgestellt wurde, wurde die Sprache in der Technologieszene praktisch über Nacht sehr populär. Grund hierfür war unter anderem die JVM. Es handelt sich hierbei um einen Interpreter für fremde Sprachen – also eine Art Dolmetscher –, mit dem Java-Bytecode in die systemeigene Sprache des jeweiligen Computers übertragen wird. Wenn Sie also eine Java-Bytecodedatei an meinen Windows-Computer übergeben, interpretiert die JVM diese Datei für die Windows-Umgebung. Übergeben Sie dieselbe Java-Bytecodedatei an den Mac meines Kollegen, dann wird derselbe Bytecode von der Macintosh-JVM für die Macintosh-Umgebung interpretiert.

Betrachten Sie noch einmal Abbildung 1.5. Ohne virtuelle Maschine müssten Sie für jedes Betriebssystem einen anderen Objektcode erstellen lassen. Dank der JVM jedoch können Sie denselben Bytecode auf Windows- oder UNIX-Computern, Macintosh-Rechnern und so weiter ausführen. Man bezeichnet dies als *Portabilität* – eine in der Welt der Computerprogrammierung sehr bequeme Eigenschaft. Denken Sie einmal an all die Leute, die auf ihren Computern im Internet unterwegs sind. Nicht alle verwenden Microsoft Windows, aber auf jedem der betreffenden Computer läuft ein eigener Bytecode-Interpreter, also eine eigene JVM.

Die Werbefachleute bei Oracle nennen dieses computertechnische Modell *Write Once, Run Anywhere*, denn Sie brauchen den Code nur einmal zu schreiben, und dann kann er überall ausgeführt werden. Das nenne ich einen tollen Ansatz zum Entwickeln von Software.

ich ein Programm zum Suchen und Ersetzen. Dann eine Rechtschreibprüfung. Als Nächstes kommt dann wieder ein Programm zum Suchen und Ersetzen. Dann eine andere Rechtschreibprüfung. Und dann eine noch bessere Lösung für das Suchen und Ersetzen von Text.«

Wie nun hat sich Chris die Begeisterung für seine Tätigkeit bewahrt? Und wie hat es das Unternehmen, bei dem er arbeitete, geschafft, im Geschäft zu bleiben? Schließlich musste Chris alle paar Monate das Rad neu erfinden, das alte Programm zum Suchen und Ersetzen wegwerfen und ein von Grund auf neues Programm entwickeln. Effizienz geht anders. Eine spannende Tätigkeit erst recht.

Jahrelang suchten Computerfachleute nach dem Heiligen Gral: einer Möglichkeit, Software so zu schreiben, dass man sie wiederverwenden kann. Die Lösung: Der Code zum Suchen und Ersetzen wird nicht immer wieder von Grund auf neu geschrieben, sondern man unterteilt die betreffende Aufgabe in kleine Einheiten. Eine Einheit sucht nach einem einzelnen Zeichen und eine zweite nach Leerzeichen, eine dritte wiederum ersetzt ein Zeichen durch ein anderes. Hat man alle Einheiten zusammen, dann bildet man aus der Summe ein Programm zum Suchen und Ersetzen. Sollte man dann später eine neue Funktion in der Textverarbeitungssoftware entwickeln, dann setzt man die einzelnen Einheiten einfach auf eine etwas andere Weise zusammen. Das ist sinnvoll, kostengünstig und vor allem nicht so langweilig.

In den späten 1980er-Jahren wurden in der Softwareentwicklung beträchtliche Fortschritte gemacht, und in den frühen 1990ern wurden bereits zahlreiche große Programmierprojekte aus vorgefertigten Komponenten erstellt. Java betrat die Bühne im Jahr 1995, und so war es für die Begründer der Sprache naheliegend, eine Bibliothek mit wiederverwendbarem Code

aufzubauen. Diese Bibliothek enthielt anfangs ca. 250 Programme, zum Beispiel Code für die Arbeit mit Dateien auf Datenträgern, Code zur Erstellung von Fenstern und Code für die Übermittlung von Daten über das Internet. Seit 1995 ist diese Bibliothek auf nun mehr als 4.000 Programme angewachsen. Ihr Name lautet *Application Programming Interface* (*API*).

Auch die allereinfachsten Java-Programme rufen Code aus der Java-API auf. Die Java-API ist gleichermaßen nützlich und komplex. Der Nutzen ergibt sich aus all den Dingen, die man mit den in ihr gespeicherten Programmen machen kann, die Komplexität hingegen aus ihrer schieren Größe. Es gibt eigentlich niemanden, der alle von der Java-API bereitgestellten Funktionen kennt. Normalerweise haben Programmierer jene Funktionen im Kopf, die sie häufig verwenden, während sie die eher selten eingesetzten nachschlagen. Dies tun sie in einem Onlinedokument namens *API-Spezifikation* (diese wird von den meisten Java-Programmierern liebevoll *API-Dokumentation* oder einfach *Javadocs* genannt).

Die unter `http://docs.oracle.com/javase/8/docs/api` zu findende API-Dokumentation beschreibt die Tausenden von Funktionen in der Java-API. Als Java-Programmierer haben Sie mit ihr tagtäglich zu tun. Sie können ein Lesezeichen auf der Oracle-Website setzen und sie immer dann besuchen, wenn Sie etwas nachschlagen müssen, oder eine Kopie der API-Dokumente von der URL `www.oracle.com/technetwork/java/javase/downloads/index.html` herunterladen.

Ihre Programmierwerkzeuge für Java

Zum Schreiben von Java-Programmen brauchen Sie die Tools, die Sie bisher in diesem Kapitel kennengelernt haben:

- ✔ **Java-Compiler.** Siehe Abschnitt »Code übersetzen«.
- ✔ **JVM.** Siehe Abschnitt »Code ausführen«.
- ✔ **Java-API.** Siehe Abschnitt »Gebrauchsfertiger Code«.
- ✔ **Zugang zur Dokumentation der Java-API.** Siehe ebenfalls Abschnitt »Gebrauchsfertiger Code«.

Hinzu kommen einige wenige exotische Tools:

- ✔ **Editor zum Erstellen von Java-Programmen.** Listing 1.1 enthält einen Auszug aus einem Computerprogramm. Im Wesentlichen handelt es sich dabei um ganz normalen Text. Deswegen brauchen Sie zum Erstellen von Computerprogrammen einen *Editor*, das heißt ein Tool zum Schreiben von Textdokumenten.

  Editoren ähneln Microsoft Word oder artverwandten Textverarbeitungsprogrammen. Der wesentliche Unterschied besteht darin, dass ein Editor keine Formatierungen zum Text hinzufügt. Es gibt also keine kursiv oder in Fettdruck gesetzten Texte und auch keine unterschiedlichen Schriftarten. Computerprogramme kennen keine Formatierung. Zu ihrer Entwicklung werden lediglich das gute alte Alphabet, Ziffern und andere Zeichen verwendet, die Sie von Ihrer Tastatur her kennen.

Wenn Sie ein Programm bearbeiten, werden zwar bestimmte Textelemente in Fett- oder Kursivdruck oder in unterschiedlichen Farben angezeigt, doch sind diese Formatierungen nicht Bestandteil des fertigen Computerprogramms. Es handelt sich hierbei vielmehr um *Syntaxhervorhebungen*. Mit ihrer Hilfe unterstützt der Editor Sie beim Nachvollziehen der Programmstruktur. Und glauben Sie mir: Die Syntaxhervorhebung ist ausgesprochen nützlich.

- ✔ **Befehlszeilenprogramm.** Sie müssen Ihrem Computer Anweisungen wie »Kompiliere dieses Programm« oder »Führe die JVM aus« mitteilen können. Jeder Computer bietet die Möglichkeit zum Absetzen solcher Befehle. (Sie führen hierzu beispielsweise Doppelklicks auf Symbole aus oder geben Befehle in ein Dialogfeld AUSFÜHREN ein.) Wenn Sie jedoch die Funktionen Ihres Computers verwenden, springen Sie zwischen den Fenstern hin und her. In dem einen Fenster ist die Java-Dokumentation geöffnet, in einem anderen bearbeiten Sie ein Java-Programm, und ein drittes dient dem Start des Java-Compilers. Diese Vorgehensweise kann sehr aufwendig sein.

Dankenswerterweise können wir auf eine Möglichkeit zurückgreifen, bei der Ihnen Programmbearbeitung, Dokumentation und Befehlsfunktion über eine einzige ansprechende Benutzeroberfläche zur Verfügung stehen. Eine solche Oberfläche nennt man *IDE* (*Integrated Development Environment*, *Integrierte Entwicklungsumgebung*).

Bei einer IDE ist das Arbeitsfenster auf dem Bildschirm normalerweise in mehrere Bereiche unterteilt: einer für die Bearbeitung von Programmen, einer zum Auflisten von Programmnamen, ein dritter zum Absetzen von Befehlen und weitere Fensterbereiche, die Sie beim Entwickeln und Testen von Programmen unterstützen. Diese Bereiche können für einen schnellen Zugriff frei angeordnet werden. Was noch besser ist: Wenn Sie Informationen in einem Bereich ändern, werden die anderen Bereiche automatisch von der IDE aktualisiert.

Mit einer IDE können Sie nahtlos zwischen den einzelnen Aufgaben bei der Programmierung wechseln. Sie brauchen sich dann nicht um die Details von Bearbeitung, Kompilierung und Ausführung der JVM zu kümmern, sondern können sich auf die Umsetzung der Programmlogik konzentrieren. (Haben Sie das nicht gewusst? Sie müssen sich auf die eine oder andere Weise immer darauf konzentrieren.)

Was schon auf Ihrer Festplatte vorhanden sein könnte

Unter Umständen befinden sich einige der Tools, die Sie zur Erstellung von Java-Programmen benötigen, bereits auf Ihrer Festplatte. Allerdings können dies bei älteren Computern durchaus veraltete Versionen sein. Die meisten Beispiele in diesem Buch können unter allen Java-Versionen ausgeführt werden. Einige Listings erfordern allerdings mindestens Java 5.0, andere womöglich Java 6, 7, 8 oder höher.

Am besten fahren Sie, wenn Sie alle Tools frisch von `java.com` oder der Oracle-Website herunterladen. Ausführliche Hinweise zum Download finden Sie in Kapitel 2.

Eclipse

Die Programme in diesem Buch funktionieren in Verbindung mit jeder IDE, die Java ausführen kann. Dazu gehören beispielsweise NetBeans, IntelliJ IDEA, JDeveloper, JCreator und

andere. Sie können die Programme sogar ohne IDE ausführen. Zur Veranschaulichung der Beispiele in diesem Buch verwende ich allerdings die IDE Eclipse. Warum ich Eclipse anderen IDEs vorziehe, hat mehrere Gründe:

- ✔ Eclipse ist kostenlos.
- ✔ Unter allen Java-IDEs ist Eclipse die unter professionellen Programmierern wohl am weitesten verbreitete.
- ✔ Zwar bringt Eclipse eine Menge Ballast mit, aber den können Sie größtenteils ignorieren. Sie müssen lediglich einige wenige Schrittabfolgen erlernen; wenn Sie Eclipse erst ein paar Mal verwendet haben, werden Sie die entsprechenden Handlungen im Schlaf ausführen können. Danach brauchen Sie keine weiteren Gedanken an Eclipse zu verschwenden, sondern können sich ganz auf die Java-Programmierung konzentrieren.