Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Marco Block

JAVA Intensivkurs

In 14 Tagen lernen
Projekte erfolgreich zu realisieren

2. Auflage

Unter Mitarbeit von
Ernesto Tapia und Felix Franke

Springer

Dr. Marco Block
Mediadesign Hochschule Berlin
Fachbereich Gamedesign und Gamedevelopment
Lindenstraße 20–25
10969 Berlin

m.block@mediadesign-fh.de
http://www.marco-block.de

ISSN 1439-5428
ISBN 978-3-642-03954-6 e-ISBN 978-3-642-03955-3
DOI 10.1007/978-3-642-03955-3
Springer Heidelberg Dordrecht London New York

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Einbandgestaltung: KünkelLopka, Heidelberg

Gedruckt auf säurefreiem Papier

Springer ist Teil der Fachverlagsgruppe Springer Science+Business Media (www.springer.de)

Für meine Katrin

Vorwort zur zweiten Auflage

Seit dem der Java-Intensivkurs als Buch erhältlich ist, habe ich viele interessante Gespräche mit Lesern und Dozenten führen können, die sich mit dem Buch auseinander gesetzt haben. Neben kleinen Fehlern, die sich bei einem Buchprojekt immer einschleichen können, wurden aber auch größere Konzepte kritisch unter die Lupe genommen. Des öfteren kam beispielsweise der Vorschlag, für das Tic-Tac-Toe-Projekt noch einen künstlichen Gegnerspieler zu implementieren, damit das Spiel sofort spielbar sei.

Besonders hat mich gefreut, dass der Autor des Javabuches „Grundkurs Java“, mit dem ich damals als Student Java gelernt habe und es später in allen meinen Veranstaltungen als Lehrbuch verwendet und empfohlen habe, Prof. Dr. Dietmar Abts, den Java-Intensivkurs ebenfalls sehr ausführlich und kritisch gelesen hat. Ihm haben die anspruchsvollen Beispiele sehr gut gefallen und mit der Zusendung einer ausführlichen Errata hat er mit dazu beigetragen, das Buch weiterzuentwickeln.

Die Webseite zum Buch, in der zunächst nur die Buchprogramme zum Download bereit stehen sollten, hat sich durch die Erweiterung um ein Forum und weitere Sparten weiter entwickelt und ist in der Zwischenzeit sogar eine eigenständige Plattform geworden. Viele Mitarbeiter und Dozenten aus Universitäten und Java-Experten aus Unternehmen helfen als ehrenamtliche Tutoren dabei, Anfängern beim Erlernen der Sprache Java unter die Arme zu greifen und größere Projekte bei der Entwicklung zu betreuen. Es gibt eine Rezensionssparte, in der aktuelle Java-Literatur aus den unterschiedlichsten Informatikbereichen vorgestellt wird.

Erfreut hat mich, dass das Buch fast immer ein sehr gutes Gesamturteil bekommen hat und gerade das hat mich sehr motiviert, nicht nur eine Kopie als zweite Auflage erscheinen zu lassen, sondern weitere Konzepte und Ideen, die ich in der Lehre sammeln konnte, einzuarbeiten.

Die zweite Auflage stellt demnach ein komplett überarbeitetes Buch dar. Die Struktur einiger Kapitel hat sich im Gegensatz zur ersten Auflage wesentlich geändert. Hinzu kamen einige neue Konzepte. Es gibt neue Aufgaben zu den entsprechenden Kapiteln mit Lösungen im Forum.

Illustrationen

Die zweite Auflage des Java-Intensivkurses wurde durch ein dreiköpfiges Designerteam der Mediadesign Hochschule Berlin durch zahlreiche Illustrationen aufgewertet. Anna Bonow hat dabei die Hauptfigur Bob und alle Zeichnungen entworfen und wurde tatkräftig durch Janina Will und Florian Häckh bei der Fertigstellung der Bilder unterstützt. Der witzige Neandertaler begleitet den Leser jetzt in allen Phasen dieses Buches und ist eine tolle Bereicherung. Jede Abbildung für sich enthält eine kleine Anekdote und ist eng mit dem Inhalt des jeweiligen Kapitels verknüpft.

Vielen Dank noch einmal an dieser Stelle für die tollen Illustrationen und die kreative Zusammenarbeit!

Unterstützung erhielt ich auch von *Sonja Rörig*, die die Insel Java aus Kapitel 1 entworfen hat.

Zusatzmaterialien und Webseite zum Buch

Neben den Beispielprogrammen und Lösungen zu den Aufgaben aus diesem Buch steht eine ständig wachsende Sammlung an kommentierten Programmen und Projekten auf der Webseite zum Buch

http://www.java-uni.de

zur Verfügung. Es gibt darüber hinaus ein Forum, in dem Fragen erörtert und Informationen ausgetauscht werden können:

http://www.java-uni.de/forum/index.php

Neben ehrenamtlichen Java-Tutoren gibt es professionelle Java-Entwickler, die im Forum unterwegs sind und gerne bei kleinen und großen Problemen helfen.

Übersicht der Kapitel

Da die Größe der einzelnen Kapitel etwas variiert, sei dem Leser angeraten, auch mal zwei kleine Kapitel an einem Tag durchzuarbeiten, wenn der Stoff keine Schwierigkeiten bereitet. An kniffligen Stellen, wie z. B. der Einführung in die Objektorientierung in Kapitel 6, kann dann mehr Zeit investiert werden. Die Erfahrung zeigt, dass der Lehrstoff dieses Buches in 14 Tagen sicher aufgenommen und erfasst werden kann.

Kapitel 1 soll die Motivation zum Selbststudium wecken, bei der Bereitstellung und Inbetriebnahme einer funktionsfähigen Java-Umgebung unterstützen und die kleinsten Java-Bausteine vorstellen. Damit wird das Fundament für das Verständnis der Programmentwicklung gelegt.

Kapitel 2 führt behutsam in die grundlegenden Prinzipien der Programmentwicklung ein und setzt diese anhand von konkreten Beispielen in Java um. Für die praktische Arbeit wird eine Klasse in Java zunächst nur als Programmrumpf interpretiert.

In Kapitel 3 werden das Ein- und Auslesen von Daten behandelt. Diese Daten können in Dateien vorliegen oder dem Programm in der Konsole übergeben werden.

In Kapitel 4 wird der Umgang mit Arrays und Matrizen durch das erste Projekt ***Conway's Game of Life*** vermittelt.

Bevor mit der Objektorientierung begonnen wird, zeigt Kapitel 5 auf, welche Regeln bei der Erstellung von Programmen zu beachten sind und mit welchen Hilfsmitteln Fehler gefunden werden können.

In Kapitel 6 wird das Klassenkonzept vorgestellt. Mit Hilfe eines ***Fußballmanagers*** wird das Konzept der Vererbung vermittelt.

Da eine Einführung in die Objektorientierung mehr als nur ein Kapitel in Anspruch nimmt, werden die bisher ausgeklammerte Fragen aus den vorhergehenden Kapiteln zum Thema Objektorientierung in Kapitel 7 aufgearbeitet.

Java verfügt im Kern über einen relativ kleinen Sprachumfang. Die Sprache lässt sich durch Bibliotheken beliebig erweitern. Kapitel 8 zeigt die Verwendung solcher Bibliotheken. Ein ***Lottoprogramm*** und das Projekt ***BlackJack*** werden mit den bisher kennengelernten Hilfsmitteln realisiert.

Kapitel 9 gibt Schritt für Schritt eine Einführung in die Erstellung von grafischen Oberflächen und die Behandlung von Fenster- und Mausereignissen.

In Kapitel 10 wird neben einer Kurzeinführung in HTML das Konzept von Applets vermittelt. Es genügen oft nur einfache Änderungen, um aus einer Applikation ein Applet zu machen.

Einen Einstieg in die Techniken der Programmentwicklung gibt Kapitel 11. Es werden viele verschiedene Programmbeispiele zu den jeweiligen Entwurfstechniken vorgestellt.

Kapitel 12 macht einen Ausflug in die Bildverarbeitung. ***Fraktale*** werden gezeichnet und verschiedene Techniken der Bildverarbeitung aufgezeigt.

Kapitel 13 beschäftigt sich mit Aspekten der Künstlichen Intelligenz, wie z. B. der ***Erkennung handgeschriebener Ziffern*** oder der Funktionsweise eines perfekt spielenden ***TicTacToe***-Spiels.

Abschließend werden in Kapitel 14 alle Phasen einer Projektentwicklung für eine Variante des Spiels ***Tetris***, vom Entwurf über die Implementierung bis hin zur Dokumentation, dargestellt.

Um den Leser für neue Projekte zu motivieren, werden in Kapitel 15 weitere Konzepte der Softwareentwicklung kurz erläutert.

Danksagungen

Auch zu der zweiten Auflage haben wieder viele Studenten und Leser ihren Teil dazu beigetragen, Fehler und Unklarheiten aufzudecken und damit das Buch zu verbessern. Ganz besonders möchte ich Miao Wang, Johannes Kulick und Benjamin Bortfeldt erwähnen, mit denen ich neben einigen Lehrveranstaltungen auch zahlreiche Java-Projekte mit Studenten gestartet habe. Durch Ihr Engagement und die kreativen Ideen werden die Studenten ermuntert, sich eigene kleine Projekte auszudenken und diese in der Gruppe zu realisieren.

Besonders möchte ich mich auch bei den vielen kritischen Lesern, die zahlreiche Korrekturen und Verbesserungen beigesteuert haben, und für die vielen bereichernden Diskussionen bedanken (in alphabetischer Reihenfolge): *Dietmar Abts*, *Maro Bader*, *Benjamin Bortfeldt*, *Anne*, *Katrin*, *Inge und Detlef Berlitz*, *Erik Cuevas*, *Jan Dérer*, *Christian Ehrlich*, *Margarita Esponda*, *Dominic Freyberg*, *Niklaas Görsch*, *Christine Gräfe*, *Ketill Gunnarson*, *Tobias Hannasky*, *Julia und Thorsten Hanssen*, *Frank Hoffmann*, *Maximilian Höflich*, *Nima Keshvari*, *Michael Kmoch*, *André Knuth*, *Raul Kompaß*, *Falko Krause*, *Jan Kretzschmar*, *Klaus Kriegel*, *Johannes Kulick*, *Tobias Losch*, *Adrian Neumann*, *Günter Pehl*, *André Rauschenbach*, *Raúl Rojas*, *Michael Schreiber*, *Bettina Selig*, *Sonja und Thilo Rörig*, *Alexander Seibert*, *Manuel Siebeneicher*, *Mark Simon*, *Ole Schulz-Trieglaff*, *Tilman Walther*, *Miao Wang*, *Daniel Werner* und *Daniel Zaldivar*.

Ich wünsche dem Leser genauso viel Spaß beim Lesen wie ich es beim Schreiben hatte und hoffe auf offene Kritik und weitere Anregungen.

Berlin, im August 2009 *Marco Block*

Aus dem Vorwort zur ersten Auflage

Es existieren viele gute Bücher, die sich mit der Programmiersprache Java auseinandersetzen und warum sollte es sinnvoll sein, ein weiteres Buch zu schreiben? Aus meiner Sicht gibt es zwei wichtige Gründe dafür.

Während der Arbeit in den Kursen mit Studenten aus verschiedenen Fachrichtungen habe ich versucht, in Gesprächen und Diskussionen herauszufinden, welche Probleme es beim Verständnis und beim Erlernen der „ersten Programmiersprache“ gab. Ein Programmierer, dem bereits Konzepte verschiedener Programmiersprachen bekannt sind, erkennt schnell die Zusammenhänge und interessiert sich primär für die syntaktische Ausprägung der neu zu erlernenden Programmiersprache. An der *Freien Universität Berlin* habe ich einige Kurse betreut, bei denen ein Großteil der Studenten keinerlei oder kaum Erfahrung im Umgang mit Programmiersprachen besaßen. Eine Motivation für dieses Buch ist es, die Erkenntnisse und Schlüsselmethoden, die ich im Laufe der Zeit gesammelt habe, auch anderen zugänglich zu machen.

Ich bin der Ansicht, dass gerade Java als Einstiegssprache besonders gut geeignet ist. Sie ist typsicher (die Bedeutung dessen wird in Kapitel 2 klar) und verfügt im Kern über einen relativ kleinen Sprachumfang. Schon in kürzester Zeit und mit entsprechender Motivation lassen sich die ersten Softwareprojekte in Java erfolgreich und zielsicher realisieren.

Dieses Buch hat nicht den Anspruch auf Vollständigkeit, dem werden andere gerecht. Es wird vielmehr auf eine Ausbildung zum Selbststudium gesetzt. Ein roter Faden, an dem sich dieses Buch orientiert, versetzt den Leser in nur 14 Tagen in die Lage, vollkommen eigenständig Programme in Java zu entwickeln. Zu den sehr vielseitigen Themengebieten gibt es viele Beispiele und Aufgaben. Die Lösungen zu den Aufgaben lassen sich im Forum auf der Buchwebseite finden. Ich habe darauf verzichtet, „Standardbeispiele“ zu verwenden und versucht, auf frische neue Anwendungen und Sichtweisen zu setzen.

Mitarbeiter dieses Buches

Motivation und Ausdauer, aus diesem über die Jahre zusammengestellten Manuskript, ein komplettes Buch zu erarbeiten, sind unter anderem *Ernesto Tapia* und *Felix Franke* zu verdanken. Beide haben nicht nur mit ihrem Engagement bei dem Aufspüren von Fehlern und Ergänzungen ihren Teil beigetragen, sondern jeweils ein Kapitel beigesteuert und damit das Buch abwechslungsreicher gestaltet.

Ernesto Tapia, der auf dem Gebiet der Künstlichen Intelligenz promoviert hat, entwarf das Tetris-Projekt für Kapitel 14 „Entwicklung einer größeren Anwendung“ und hatte großen Einfluss auf Kapitel 13 „Methoden der Künstlichen Intelligenz“. Für die seit vielen Jahren in Forschung und Lehre währende gemeinsame Arbeit und Freundschaft bin ich ihm sehr dankbar.

Felix Franke war vor Jahren selbst einer meiner Studenten, dem ich den Umgang mit Java vermittelte. Durch seine Zielstrebigkeit konnte er sein Studium vorzeitig beenden und promoviert ebenfalls auf dem Gebiet der Künstlichen Intelligenz. Er hat sich voll und ganz dem Kapitel 12 „Bildverarbeitung“ gewidmet.

Danksagungen

Mein Dank gilt allen, die auf die eine oder andere Weise zur Entstehung dieses Buches beigetragen haben. Dazu zählen nicht nur die Studenten, die sichtlich motiviert waren und mir Freude beim Vermitteln des Lehrstoffs bereiteten, sondern auch diejenigen Studenten, die im Umgang mit Java große Schwierigkeiten hatten. Dadurch wurde ich angeregt auch unkonventionelle Wege auszuprobieren.

Die Vorlesungen an der Freien Universität Berlin von *Klaus Kriegel*, *Frank Hoffmann* und *Raúl Rojas* haben mein Verständnis für didaktische Methodik dabei am intensivsten geprägt.

An dieser Stelle möchte ich meinen Eltern, Großeltern und meinem Bruder danken, die mich immer bei allen meinen Projekten bedingungslos unterstützen.

Unterstützung erhielt ich von *Sonja Rörig*, die neben der Arbeit als Illustratorin und Designerin, Zeit für meine Abbildungen fand. Das Vererbungsbeispiel mit den witzigen Fußballspielern hat es mir besonders angetan. Vielen lieben Dank für Deine professionelle Hilfe! Auch *Tobias Losch* ist in diesem Zusammenhang noch einmal zu nennen, denn neben den zahlreichen Korrekturhilfen, hat auch er bei einigen Abbildungen geholfen und die Webseite zum Buch entworfen.

Die Zusammenarbeit mit dem Springer-Verlag, gerade bei diesem ersten Buchprojekt, war sehr angenehm und bereichernd. Für die geduldige und fachkundige Betreuung von *Hermann Engesser* und *Gabi Fischer* bin ich sehr dankbar.

Berlin, im Juli 2007 *Marco Block*

Inhaltsverzeichnis

1

Tag 1: Vorbereitungen und Javas kleinste Bausteine

Unser erster Abschnitt beschäftigt sich zunächst mit der Installation von Java, der Wahl einer Entwicklungsumgebung und der Motivation zum Selbststudium. Im zweiten Teil dieses Kapitels werden wir die kleinsten Java-Bausteine kennenlernen und schon mit den ersten Programmierübungen beginnen.

Ein innerer Antrieb und viele praktische Übungen sind unerlässlich für das Erlernen einer Programmiersprache. Programmieren heißt häufig auch, mit anderen oder für andere zu programmieren. Daher werden wir lernen, durch einen verständlichen

M. Block, *JAVA-Intensivkurs*
DOI 10.1007/978-3-642-03955-3, © Springer 2010

Programmaufbau, mit Diagrammen und ausreichender Kommentierung, die Kommunikation mit anderen zu verbessern, um allein oder gemeinsam komplexe Projekte meistern zu können.

1.1 Warum gerade mit Java beginnen?

Es gibt viele Gründe mit dem Programmieren zu beginnen, aber warum sollte es gerade die Programmiersprache Java sein? Eine Insel in Indonesien trägt ebenfalls den Namen Java (siehe Abb. 1.1).

Die Vorteile von Java liegen in dem, im Gegensatz zu anderen Programmiersprachen, relativ kleinen Befehlsumfang und der strikten Typsicherheit. Java ist plattformunabhängig und kann daher auf verschiedenen Betriebssystemen eingesetzt werden. Auf mobilen Geräten hat Java in der Zwischenzeit eine dominierende Stellung eingenommen. Es gibt eine sehr große Java-Community und die Sprache ist leicht zu erlernen. Generell gilt: Die Chance in Java etwas falsch zu machen ist sehr viel kleiner als z. B. bei C++.

Diese Eigenschaften sind der Grund dafür, weshalb Java gerade in der Lehre intensiv eingesetzt wird.

Es gibt aber auch Nachteile. Geschwindigkeitsrelevante Softwaresysteme sollten nicht mit Java, sondern eher mit Programmiersprachen wie C oder C++ geschrieben werden. Trotzdem hat sich Java gerade in der Mobilkommunikation durchgesetzt. Der Speicher eines Computers kann nicht direkt angesprochen werden, alles wird über eine virtuelle Maschine gesteuert. Damit ist auch sichergestellt, dass sicherheitsproblematische Speicherabschnitte nicht so einfach ausgelesen werden können (siehe dazu Abschn. 1.3.9 in [40]).

Neben den üblichen **Applikationen**, das sind selbstständige Anwendungsprogramme (stand-alone Applications), können wir mit Java aber auch Programme schreiben, die in eine Webseite eingebunden werden können. Diese Programme nennen wir **Applets**.

In diesem Buch werden wir beide Programmtypen kennenlernen und mit ihnen verschiedene Projekte realisieren. Wir werden sehen, dass oft nur wenige Arbeitsschritte notwendig sind, um eine Applikation in ein Applet zu ändern und damit webfähig zu machen.

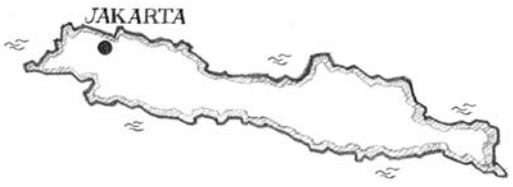

Abb. 1.1. Java ist auch eine indonesische Insel mit der Hauptstadt Jakarta

1.2 Installation von Java

Wir beginnen zunächst damit, uns mit dem System vertraut zu machen. Dazu installieren wir auf dem vorhandenen Betriebssystem eine aktuelle Java-Version. Für das vorliegende Buch wurde die Version 1.6 verwendet. Es ist darauf zu achten, dass es sich bei dieser um eine SDK- oder JDK-Version handelt (SDK = Software Development Kit oder JDK = Java Development Kit). Zu finden ist sie zum Beispiel auf der Internetseite von Sun Microsystems [53].

Eine wichtige Anmerkung an dieser Stelle: Bei vielen Betriebssystemen ist es notwendig, die **Umgebungsvariablen** richtig zu setzen. Bei Windows XP beispielsweise geht man dazu in die „Systemsteuerung“ und dort auf „Systemeigenschaften“, dann müssen unter der Rubrik „Erweitert“ die Umgebungsvariablen geändert werden. Es gibt eine Variable *PATH*, die um den Installationsordner + „`/bin`“ von Java erweitert werden muss. In älteren Systemen muss noch eine neue Umgebungsvariable mit dem Namen *CLASSPATH* erzeugt und ihr ebenfalls der Javapfad zugewiesen werden. Zusätzlich zum Javapfad muss „`;.;`“ angehängt werden.

Da sich Betriebssysteme untereinander und sogar von Version zu Version stark im Verhalten unterscheiden können, bietet es sich bei auftretenden Schwierigkeiten an, das Java-Forum zu besuchen. Dort lassen sich bereits veröffentlichte Lösungen nachlesen oder wir werden eine Lösung gemeinsam finden.

1.2.1 Wahl einer Entwicklungsumgebung

Nach der Installation von Java müssen wir uns für eine Entwicklungsumgebung entscheiden und diese ebenfalls installieren. Es gibt eine Reihe von Umgebungen, die es dem Programmierer erleichtern, Programme in Java zu entwickeln. Hier sind ein paar kostenfrei erhältliche Programme aufgelistet:

- Jeder Texteditor kann verwendet werden, z. B. NotePad++ [58]
- Eclipse [55]
- JCreator [56]
- NetBeans [57]
- Borland JBuilder [59]

Es gibt viele weitere Editoren für Java. Da ich keinen bevorzuge, sondern den angehenden Programmierer für die internen Prozesse sensibilisieren möchte, verzichte ich auf den Einsatz professioneller Entwicklungsumgebungen und schreibe alle Programme mit einem einfachen Texteditor (siehe Abb. 1.2).

Dem Leser sei es selbst überlassen, eine Wahl zu treffen. Eine Übersicht findet sich beispielsweise auf der Internetseite vom Java-Tutor [54].

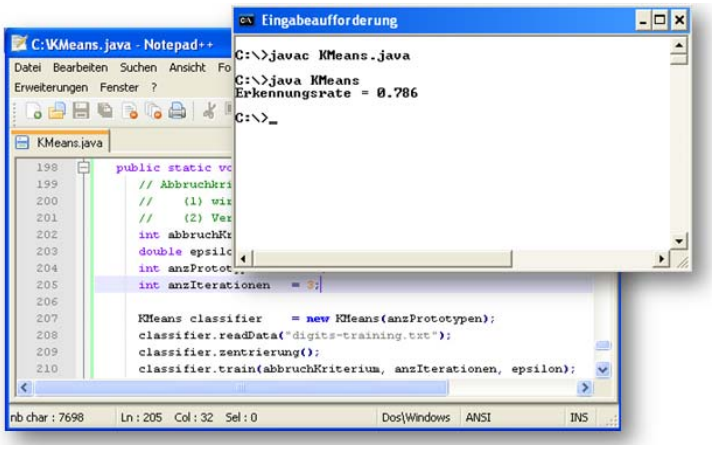

Abb. 1.2. Javaprogramme entwickeln mit NotePad++

Für Umsteiger, die bereits Erfahrung mit der Entwicklung von Programmen in anderen Sprachen besitzen, empfehle ich Eclipse. Anfänger sollten mit dem Einsatz von Eclipse erst beginnen, wenn Ihnen die Entwicklung von Javaprogrammen keine allzu große Schwierigkeiten mehr bereitet. Die Fülle an Funktionen und Möglichkeiten, die von der Entwicklungsumgebung geboten wird, können schnell zu großem Frust führen, wenn die Übersicht verloren geht und die einfachsten Fehler nicht mehr zu lösen sind.

1.2.2 Testen wir das installierte Java-System

Bevor es mit der Programmierung losgeht, wollen wir nach der Installation von Java das System auf Herz und Nieren prüfen. Dazu starten wir eine **Konsole** (unter Windows heißt dieses Programm **Eingabeaufforderung**). Wenn die Installation von Java vollständig abgeschlossen ist und die Umgebungsvariablen richtig gesetzt sind, dann müsste die Eingabe der Anweisung *javac* zu folgender (hier gekürzter) Ausgabe führen:

```
C:\>javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler ...
  -deprecation               Output source locations where deprecate ...
  ...
```

In dieser oder ähnlicher Form sollte die Ausgabe geliefert werden. Falls eine Fehlermeldung erscheint, z. B.

```
C:\>javac
Der Befehl "javac" ist entweder falsch geschrieben oder
konnte nicht gefunden werden.
```

bedeutet dies, dass das Programm *javac* nicht gefunden wurde. An dieser Stelle sollte vielleicht die *PATH*-Variable noch einmal überprüft (siehe dazu Abschn. 1.2) oder ein Rechnerneustart vorgenommen werden.

Testen wir unser System weiter. Um herauszufinden, ob der *CLASSPATH* richtig gesetzt ist, schreiben wir ein kurzes Programm. Dazu öffnen wir einen Editor und schreiben folgende Zeile hinein:

```
1 public class Test{}
```

Diese Datei speichern wir mit dem Namen **Test.java** in einem beliebigen Ordner. Am besten wäre an dieser Stelle vielleicht, die Datei einfach auf „`C:\`“ zu speichern, denn wir werden das Programm anschließend sowieso wieder löschen. In meinem Fall habe ich einen Ordner mit dem Namen „`Java`“ in „`C:\`“ angelegt und die Datei dort hinein gespeichert.

Anschließend navigieren wir in der Konsole zu der Datei und geben die folgende Anweisung ein:

```
C:\>cd Java
C:\Java>javac Test.java

C:\Java>
```

Wenn keine Fehlermeldung erscheint, ist das System bereit und wir können endlich mit der Programmierung beginnen.

Dieses kleine Beispiel hat uns einen ersten Einblick in die Programmerstellung mit Java gegeben. Wir werden in einem Editor die Programme schreiben und sie anschließend in der Konsole starten. Mit der Anweisung *javac* wird das Programm in den so genannten Byte-Code umgewandelt, dabei wird eine Datei mit dem gleichen Namen erzeugt, aber mit der Endung „`.class`“ versehen.

Eine solche Datei wird auch für unser kleines Beispiel erzeugt.

```
C:\Java>javac Test.java

C:\Java>dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: A8DB-F3AF

 Verzeichnis von C:\Java

14.03.2007  12:22    <DIR>          .
14.03.2007  12:22    <DIR>          ..
14.03.2007  12:22               182 Test.class
14.03.2007  12:10                19 Test.java
               2 Datei(en)            201 Bytes
               2 Verzeichnis(se), 372.468.928.512 Bytes frei
```

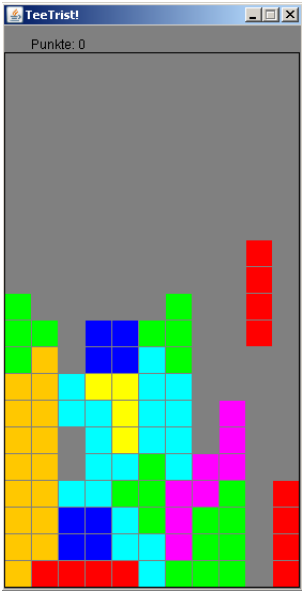

Abb. 1.3. Das Tetrisprojekt in Kap. 14 besitzt eine grafische Oberfläche. Über Tastatureingaben lassen sich die herabfallenden Steine bewegen und rotieren

Später werden wir noch erfahren, wie wir einen solchen Byte-Code starten und beispielsweise ein kleines Tetrisprogramm spielen können (siehe Abb. 1.3).

1.3 Vorteile des Selbststudiums

Es gibt zwei Ansätze beim Erlernen einer Programmiersprache. Der erste Ansatz ist ein detailliertes Studium aller Funktionen und Eigenschaften einer Sprache und das fleißige Erlernen aller Vokabeln. Dabei können zu jedem Teilaspekt Beispiele angebracht und Programmierübungen gelöst werden. Diese Strategie könnte als ***Bottom-Up-Lernen*** bezeichnet werden, da erst die kleinen Bausteine gelernt und später zu großen Projekten zusammengefügt werden.

Beim zweiten Ansatz werden die zu erlernenden Bausteine auf ein Minimum reduziert. Es werden eher allgemeine Konzepte als speziell in einer Sprache existierende Befehle und Funktionen vermittelt. Mit diesem Basiswissen und verschiedenen Wissensquellen, die zu Rate gezogen werden können, lassen sich nun ebenfalls erfolgreich größere Projekte realisieren. Die entsprechenden Bausteine werden während des Entwicklungsprozesses studiert und erarbeitet. Das zweite Verfahren, wir könnten es ***Top-Down-Lernen*** nennen, eignet sich für Personen, die sich nicht auf eine spezielle Programmiersprache beschränken und schnell mit dem Programmieren beginnen wollen. Der Lernprozess findet dabei durch die praktische Handhabung der Sprache statt.

Beim Erlernen einer Fremdsprache verhält es sich ähnlich. So kann zunächst damit begonnen werden, möglichst viele Vokabeln und die Grammatik zu lernen und anschließend in entsprechenden Situationen die Worte zu Sätzen zu verknüpfen. Der Lernprozess findet also primär vor der praktischen Ausübung statt. Bei der Top-Down-Variante wird im Gegensatz dazu zunächst das Szenario vorgegeben und nur die dafür notwendigen Vokabeln und Grammatik werden besprochen. Das Sprechen beginnt früher, Hemmungen werden abgelegt, mehr Sinne werden verwendet und der Lernprozess verlagert sich in die praktische Ausübung.

Ich persönlich bevorzuge die Top-Down-Variante und habe mit Bedacht diesen didaktischen Weg für die Konzeption dieses Buches gewählt. Aus diesem Grund wird der Leser schnell mit dem Programmieren beginnen können, ohne erst viele Seiten Theorie studieren zu müssen. Da dieses Buch nicht den Anspruch auf Vollständigkeit hat, wird dem Leser im Laufe der Kapitel vermittelt, wie und wo Informationsquellen zu finden sind und wie diese zu verwenden sind, um spezielle Probleme eigenständig zu lösen.

Es ist unmöglich, ein Buch zu schreiben, das auf alle erdenklichen Bedürfnisse eingeht und keine Fragen offen lässt. Deshalb ist es ratsam, zu entsprechenden Themenbereichen zusätzliche Quellen zu Rate zu ziehen, aus denen an vielen Stellen verwiesen wird. Dieses Buch eignet sich nicht nur als Einstieg in die Programmierung mit Java, sondern versucht auch einen Blick auf den ganzen Prozess einer Softwareentwicklung zu geben.

Java ist nur ein Werkzeug zur Formulierung von Programmen. Zur erfolgreichen Realisierung von Projekten gehört aber noch viel mehr. Projekte müssen genau geplant und vor der eigentlichen Entwicklung möglichst so formuliert werden, dass jeder Softwareentwickler weiß, was er zu tun hat.

Im Laufe der vielen Projekte im Buch durchstreifen wir verschiedene Gebiete der Informatik, vom Algorithmenentwurf, über Verfahren aus der Künstlichen Intelligenz bis hin zur Entwicklung kleiner Spiele.

Dieses Buch soll dem Leser einen umfassenden Einstieg in den gesamten Prozess der Softwareentwicklung ermöglichen.

Ich hoffe, dass Motivation und Kreativität für die Realisierung von Projekten in Java jetzt ausreichend vorhanden sind. Wir werden in den folgenden Abschnitten mit den kleinsten Bausteinen beginnen.

1.4 Primitive Datentypen und ihre Wertebereiche

Um einen ersten Einstieg in die Programmierung zu wagen und ein Fundament zu legen, müssen wir uns zunächst mit den primitiven Datentypen und den allgemeinen Prinzipien der Programmentwicklung vertraut machen. In Java wird im Gegensatz zu anderen Programmiersprachen, wie z. B. C oder C++, besonderer Wert auf Typsicherheit gelegt. Damit ist gemeint, dass der Interpreter, der das Programm ausführt,

bei der Verwaltung von Speicher immer genau Bescheid wissen möchte, um welchen Datentyp es sich handelt.

Mit Programmen wollen wir den Computer dazu bringen, bestimmte Aufgaben und Probleme für uns zu lösen. Wir haben es dabei immer mit Daten zu tun, die entweder als Ein- oder Ausgabe für die Lösung einer Aufgabe oder zum Speichern von Zwischenlösungen verwendet werden.

1.4.1 Primitive Datentypen allgemein

Die vielen Jahre der Softwareentwicklung haben gezeigt, dass es drei wichtige Kategorien von Daten gibt, die wir als atomar bezeichnen können, da sie die kleinsten Bausteine eines Programms darstellen und wir sie später zu größeren Elementen zusammenfassen werden.

Wahrheitswerte

Wahrheitswerte repräsentieren die kleinste Einheit im Rechner. Auf 0 oder 1, *true* (wahr) oder *false* (falsch), Strom *an* oder *aus* beruhen alle technischen Ereignisse eines Rechners. In jedem Programm werden sie ob bewusst oder unbewusst verwendet.

Zeichen und Symbole

Die Ein- und Ausgaben von Text sind ebenfalls wichtige Hilfsmittel für einen Programmierer. Zeichen oder Symbole, wie z. B. ‚`a`‘, ‚`3`‘, ‚`$`‘ oder ‚`)`‘, lassen sich später zu größeren Datentypen (Zeichenketten) zusammenfassen, beispielsweise zu „`Ich bin eine Zeichenkette!`“.

Zahlen

Zahlen sind wichtig, um beispielsweise Ereignisse zu zählen. Je nach Verwendungszweck und Wertebereich, z. B. -1, 5,3, 2 390 682 395 724 oder 0,293, beanspruchen Zahlen mal mehr und mal weniger Speicher im Rechner. Eine Zahl mit vielen Stellen nach dem Komma (Gleitkommazahl), von der eine große Genauigkeit erwartet wird, benötigt mehr Speicher als eine ganze Zahl ohne Nachkommastellen, die nur einen kleinen Zahlenbereich abdecken muss.

1.4.2 Primitive Datentypen in Java

Je nach Programmiersprache werden nun verschiedene Datentypen aus diesen drei Kategorien angeboten. Wir nennen sie, da sie die kleinsten Bausteine sind, primitive Datentypen.

In Java gibt es:

Wahrheitswerte: `boolean`
Zeichen und Symbole: `char`
Zahlen: `byte`, `short`, `int`, `long`, `float`, `double`

Wie diese Übersicht zeigt, wird in der Programmierung dem Bereich der Zahlen eine besondere Aufmerksamkeit gewidmet. Das hat gute Gründe. Um gute, schnelle Programme zu schreiben, ist es notwendig, sich Gedanken über die verwendeten Datentypen zu machen. Sicherlich könnte der Leser der Meinung sein, dass ein Zahlentyp ausreicht, beispielsweise ein `double`, der sowohl positive und negative als auch ganze und gebrochene Zahlen darzustellen vermag. Das ist korrekt. Aber ein `double` benötigt im Rechner jede Menge Speicher und für Operationen, wie Addition, Subtraktion, Multiplikation und Division, einen größeren Zeitaufwand als z. B. ein `short`.

Da wir in Programmen meistens sehr viele Datentypen und Operationen auf diesen verwenden wollen, um Aufgaben zu lösen, gilt die Devise, immer den Datentyp zu verwenden, der für die Aufgabe geeignet ist und den kleinsten Speicherbedarf hat.

Um nun entscheiden zu können, welche Datentypen in Frage kommen, bleibt es uns nicht erspart, einmal einen Blick auf die Wertebereiche zu werfen (Tabelle 1.1).

Diese Tabelle muss jetzt nicht auswendig gelernt werden, es genügt zu wissen, wo sie zu finden ist. Von einer Programmiersprache zu einer anderen, selbst bei verschiedenen Versionen einer Programmiersprache, können sich diese Werte unterscheiden, oder es kommen neue primitive Datentypen hinzu. Das ist unter anderem durch die enorme Weiterentwicklung der vorhandenen Prozessoren und Rechnergenerationen begründet.

Dem Leser mag aufgefallen sein, dass für einen `boolean`, obwohl er nur zwei Zustände besitzt, acht BIT reserviert sind. Logischerweise benötigt ein `boolean` nur ein BIT zur Speicherung der zwei möglichen Zustände `true` und `false`, aber aus technischen Gründen sind acht BIT die kleinste Speichereinheit in der aktuellen Rechnergeneration.

Tabelle 1.1. Übersicht der primitiven Datentypen in Java [41]. Die dritte Spalte zeigt den benötigten Speicher in BIT

Datentyp	Wertebereich	BIT
boolean	true, false	8
char	0 … 65 535 (z. B. ‚a‘, ‚b‘, … , ‚A‘, … ,‚1‘, …)	16
byte	–128 bis 127	8
short	–32 768 bis 32 767	16
int	–2 147 483 648 bis 2 147 483 647	32
long	–9 223 372 036 854 775 808 bis 9 223 372 036 854 775 807	64
float	+/–1,4E-45 bis +/–3,4E+38	32
double	+/–4,9E-324 bis +/–1,7E+308	64

1.5 Variablen und Konstanten

Mit Variablen und Konstanten haben wir erst die Möglichkeit, die Datentypen in unseren Programmen mit Inhalten zu füllen.

1.5.1 Deklaration von Variablen

Damit wir die unterschiedlichen Datentypen aus dem vorhergehenden Abschnitt in unseren Programmen einbauen können, müssen wir, wie es z. B. in der Algebra üblich ist, **Variablen** verwenden. Diese Variablen dienen als Platzhalter für die Werte im Speicher und belegen je nach Datentyp mehr oder weniger Ressourcen.

Um eine Variable eines bestimmten Typs im Speicher anzulegen, die für den Rest des Programms oder bis sie gelöscht wird bestehen bleibt, müssen wir sie **deklarieren**.

```
<Datentyp> <Name>;
```

Dabei ist `<Datentyp>` ein Platzhalter für einen Datentyp und `<Name>` ein Platzhalter für die Variablenbezeichnung. Beispielsweise wollen wir eine Variable vom Typ Wahrheitswert verwenden:

```
boolean a;
```

Die Variable `a` steht nach der **Deklaration** bereit und kann einen Wert zugewiesen bekommen. Das nennen wir eine **Zuweisung**.

Wir könnten nun nacheinander Variablen deklarieren und ihnen anschließend einen Wert zuweisen. Es lassen sich auch mehrere Variablen eines Datentyps gleichzeitig deklarieren oder, wie es die letzte Zeile demonstriert, Deklaration und Zuweisung in einer Anweisung ausführen.

```
boolean a;
a = true;
int b;
b = 7;
float c, d, e;
char f = 'b';
```

Die Bezeichnung einer Variable innerhalb eines Programms wird uns überlassen. Es gibt zwar ein paar Beschränkungen für die Namensgebung, aber der wichtigste Grund für die Wahl ist die Aufgabe der Variable.

Dazu ein paar Beispiele:

```
boolean istFertig;
double kursWert;
int schrittZaehler;
```