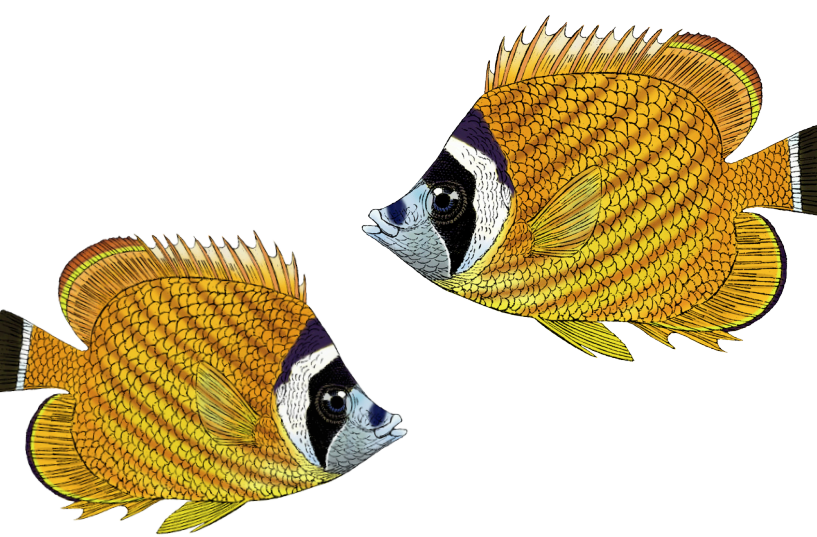
O'REILLY®

Java lernen

kurz & gut

O'Reillys Taschenbibliothek



Michael Inden

Vorwort

Zunächst einmal bedanke ich mich bei Ihnen, dass Sie sich für dieses Buch entschieden haben. Hierin finden Sie einen fundierten und interaktiven Einstieg in die Programmierung mit Java. Das Ganze startet bei den Grundlagen und basierend darauf wird Ihr Wissen immer weiter ausgebaut, sodass Sie nach der Lektüre bereit sind, eigene Experimente zu wagen, und bestenfalls Programmieren als neues Hobby lieben gelernt haben. Insbesondere die ungeheuren Möglichkeiten, kreativ zu werden und dabei immer wieder Neues zu entdecken, werden Sie bestimmt ähnlich faszinieren wie mich seit über 30 Jahren.

Zielgruppe

Dieses Buch richtet sich an Programmierneulinge und wendet sich somit insbesondere an

- **Schüler und Schülerinnen**, die ein paar Tipps und Hilfestellungen suchen, die das Nachvollziehen des Informatikunterrichts erleichtern,
- **Studierende**, die ergänzende Erklärungen zu denen aus den Vorlesungen suchen, um Gelerntes schneller anwenden zu können oder besser für die nächste Prüfung vorbereitet zu sein,
- und **alle Interessierten**, die einfach die wunderbare und vielfältige Welt der Programmierung mit Java kennenlernen möchten.

Voraussetzungen

Zum Einstieg sind Programmiererfahrungen keine zwingende Voraussetzung – natürlich schaden diese nicht. Selbst dann

nicht, wenn Sie sich vielleicht eher mit Python, C#, TypeScript oder JavaScript beschäftigt haben. Für die Lektüre des Buchs ist es aber hilfreich, wenn Sie

- einigermaßen fit im Installieren von Programmen sind und
- wissen, was die Kommandozeile ist und sie grundlegend bedienen können.

Was vermittelt dieses Buch?

Sie als Leser erhalten in diesem Buch einen Einstieg in Java. Allerdings ist die trockene Theorie auf ein Minimum reduziert und wir legen immer mit kleinen Beispielen los. Deshalb ist es ein Buch zum Mitmachen. Ich ermutige Sie, parallel zum Lesen auch immer ein paar Dinge auszuprobieren, vielleicht sogar mal das eine oder andere abzuwandeln. Man lernt Programmieren einfach am besten, wenn man es praktiziert. Somit bietet es sich an, die abgebildeten Codeschnipsel abzutippen. Das kann in der Konsolenapplikation JShell erfolgen oder im Editor Ihrer Entwicklungsumgebung/IDE (Integrated Development Environment). Beide Varianten werden im Verlauf des Buchs genauer beschrieben.

Damit Sie nicht über einfache Probleme stolpern, führt das Buch jeweils behutsam und schrittweise in die jeweilige Thematik ein und gibt Ihnen immer auch ein paar Hinweise, auf was man achten oder was man vielleicht sogar vermeiden sollte. Dazu dienen diverse Hinweise mit Hintergrundinformationen.

Hinweis: Hintergrundinformation

In derart formatierten Kästen finden sich im späteren Verlauf des Buchs immer wieder einige wissenswerte Tipps und ergänzende Hinweise zum eigentlichen Text.

Aufbau dieses Buchs

Dieses Buch besteht aus jeweils in sich abgeschlossenen, aber aufeinander aufbauenden Kapiteln zu elementar wichtigen Bereichen der Programmiersprache Java. Für Ihren erfolgreichen Weg zur Java-Programmierung gliedert sich das Buch in die beiden Teile Einstieg und Aufstieg.

Im Teil »Einstieg« werden Grundlagen behandelt. Hier empfiehlt es sich, die Kapitel in der Reihenfolge des Buchs zu lesen, da mit jedem Kapitel neue Wissensbausteine und Themen hinzukommen, die im Anschluss vorausgesetzt und verwendet werden.

Dann folgt der Teil »Aufstieg«. Dort beschäftigen wir uns mit leicht fortgeschrittenen Themen. Hier können Sie zwar nach Lust und Laune eins der Kapitel zur Lektüre auswählen, aber auch hier bauen einige Themen aufeinander auf.

Einstieg

Kapitel 1 – Einführung Dieses Kapitel gibt einen Überblick über Javas mittlerweile über 25-jährige Geschichte. Zudem wird die Installationen von Java an sich und einer Entwicklungsumgebung/IDE sowie das Ausprobieren der Beispiele beschrieben.

Kapitel 2 – Schnelleinstieg Dieses Kapitel bietet einen Schnelleinstieg und stellt viele wesentliche Elemente von Java vor. Dabei wird behutsam Fahrt aufgenommen: Wir beginnen mit einer Ausgabe eines Textes, traditionell »Hello World«, und lernen dann, wie wir das mithilfe von Variablen variieren. Zudem schauen wir uns bedingte Ausführungen mit Fallunterscheidungen sowie Wiederholungen mit Schleifen an.

Kapitel 3 – Strings Variablen vom Typ `String` repräsentieren Zeichenketten und dienen zur Verwaltung von textuellen Informationen. Diese sind aus kaum einem Programm wegzudenken und werden in diesem Kapitel genauer behandelt.

Kapitel 4 – Arrays Ebenso wie Strings sind auch Arrays recht gebräuchliche Datenstrukturen und helfen dabei, mehrere gleichartige Dinge zu speichern, etwa eine Menge von Zahlen, Namen, Personen usw. Insbesondere bilden Arrays die Grundlage für viele andere Datenstrukturen. In diesem Kapitel lernen wir Arrays im Detail kennen.

Kapitel 5 – Klassen und Objektorientierung Immer wieder hört man, Java ist eine objektorientierte Sprache. Doch was bedeutet das? Zum Verständnis gibt dieses Kapitel einen Einblick in den objektorientierten Entwurf von Software. Dazu vermittle ich die grundlegenden Ideen von Zustand (Daten) in Kombination mit Verhalten (Funktionen auf diesen Daten) und wie man dies in Java formuliert.

Kapitel 6 – Collections Während Arrays ziemlich elementar sind, bieten die in Java integrierten Collections oder Containerklassen mehr Flexibilität und Komfort bei der Verwaltung von Daten. Insbesondere unterstützen die vordefinierten Listen, Mengen und Schlüssel-Wert-Abbildungen bei der Verwaltung anderer Objekte.

Kapitel 7 – Ergänzendes Wissen In diesem Kapitel werden Themen angesprochen, die zuvor aus didaktischen Gründen bewusst ausgelassen wurden, weil das Ganze sonst zu tief in die Details gegangen wäre und zu viel anderes Wissen vorausgesetzt hätte. Hier angelangt lohnt es sich, Ihre Java-Kenntnisse zu primitiven Typen, dem Ternary-Operator, Fallunterscheidungen mit `switch` usw. zu komplettieren.

Aufstieg

Kapitel 8 – Lambdas und Streams Dieses Kapitel stellt sowohl Lambda-Ausdrücke (kurz Lambdas) als auch das damit eng verbundene Stream-API vor. Beides sind essenzielle Bausteine von modernem Java und ermöglichen es, Lösungen oftmals elegant zu formulieren.

Kapitel 9 – Verarbeitung von Dateien Dieses Kapitel beschäftigt sich mit der Verarbeitung von Informationen aus Dateien. Dies ist für viele Anwendungen von großer Bedeutung, da Informationen oftmals nicht nur während der Programmlaufzeit von Interesse sind, sondern vor allem auch darüber hinaus – denken Sie etwa an die Highscore-Liste Ihres Lieblingsspiels.

Kapitel 10 – Fehlerbehandlung mit Exceptions Vielleicht kennen Sie es schon: Manchmal tritt ein Programmfehler auf und das Programm stürzt ab. Wichtige Daten gehen potenziell verloren. So etwas ist ärgerlich. Daher gehört auch die Behandlung von Fehlern zum guten Ton beim Programmieren. Dieses Kapitel führt in die Thematik ein.

Kapitel 11 – Datumsverarbeitung Während früher die Datumsverarbeitung eher stiefmütterlich in Java unterstützt wurde, bietet modernes Java eine Vielzahl praktischer Funktionalitäten zur Datumsverarbeitung, die in diesem Kapitel einführend dargestellt werden.

Kapitel 12 – Schlusswort Hier rekapitulieren wir kurz, was Sie durch die Lektüre dieses Buchs gelernt haben sollten und wie Sie möglicherweise weitermachen können.

Anhang

Anhang A – Schlüsselwörter im Überblick In Java existiert eine Reihe von Schlüsselwörtern, die reserviert sind und nicht als Bezeichner für Variablen, Methoden, Klassen oder anderes verwendet werden dürfen. Hier erhalten Sie einen Überblick.

Anhang B – Schnelleinstieg JShell In diesem Buch werden diverse Beispiele direkt auf der Konsole ausprobiert. Dabei hilft die interaktive Kommandozeilenapplikation JShell als REPL (Read-Eval-Print-Loop).

Anhang C – Grundlagen zur JVM In diesem Anhang vermittle ich Grundwissen zur JVM (Java Virtual Machine).

Sourcecode und ausführbare Programme

Beim Erlernen des Programmierens ist es hilfreich, Beispiele selbst auszuprobieren und abzutippen. Um Ihnen ein wenig Tipparbeit und Mühe zu ersparen, finden Sie viele der Beispiele als Programme in einem Eclipse-Projekt. Dieses steht unter `https://oreilly.de/produkt/java-lernen-kurz-gut/` zur Verfügung. Weitere Informationen zum genauen Vorgehen finden Sie auf der Download-Seite.

Verwendete Java-Version(en)

Viele Beispiele wurden auf Basis von Java 17 entwickelt und ausprobiert. Mittlerweile ist bereits Java 21 erschienen. Hilfreiche Features dieser Version setze ich passend ein. Für dieses Buch sind die brandaktuellen Java-Features zwar von Interesse, aber nicht von entscheidender Bedeutung, da es ja um die Grundlagen der Sprache geht.

Nach der Lektüre dieses Buchs sind Sie bestens gerüstet für modernes Java und können nach Lust und Laune eigene Experimente und Hobbyprojekte starten.

Konventionen

Verwendete Zeichensätze

Neben der vorliegenden Schriftart sind wichtige Textpassagen *kursiv* oder ***kursiv und fett*** markiert. Englische Fachbegriffe werden eingedeutscht großgeschrieben, etwa Event Handling. Zusammensetzungen aus englischen und deutschen (oder eingedeutschten) Begriffen werden mit Bindestrich verbunden, z. B. Plugin-Manager. Listings mit Sourcecode sind in der Schrift `Courier` gesetzt, um zu verdeutlichen, dass diese einen Ausschnitt aus einem Java-Programm darstellen. Auch im normalen Text wird für Klassen, Methoden, Konstanten und Parameter diese Schriftart genutzt.

Schreibweise von Methodenaufrufen

Im Text beschriebene Methodenaufrufe enthalten in der Regel die Typen der Übergabeparameter, etwa `substring(int, int)`. Sind die Parameter in einem Kontext nicht entscheidend, wird mitunter auf deren Angabe aus Gründen der besseren Lesbarkeit verzichtet.

Verwendete Abkürzungen

Im Buch verwende ich die in der nachfolgenden Tabelle aufgelisteten Abkürzungen. Weitere Abkürzungen werden im laufenden Text in Klammern nach ihrer ersten Definition aufgeführt und anschließend bei Bedarf genutzt.

Abkürzung	Bedeutung
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
IDE	Integrated Development Environment
JDK	Java Development Kit
JEP	JDK Enhancement Proposal
JLS	Java Language Specification
JRE	Java Runtime Environment
JVM	Java Virtual Machine

Gerne möchte ich noch den Begriff API (Application Programming Interface) kurz erklären: Dabei handelt es sich um die von einem Programm bereitgestellten Funktionalitäten in Form von Methoden.

Danksagung

Dieses Buch basiert auf meinem Buch »Einfach Java« [3], das in weiten Teilen leicht überarbeitet wurde und somit ein Facelift erfahren hat. Dazu wurden diverse Passagen und Beispiele ergänzt und derart modifiziert, dass sie noch besser verständlich sind. Auch potenziell missverständliche Formulierungen

wurden korrigiert und sogar kleinere Teile entfernt, um dem Text mehr Stringenz zu verleihen und sich in die Kurz-und-gut-Serie einzupassen. Aufgrund der Basis »Einfach Java« möchte ich hier nochmals folgenden Leuten danken: Michael Kulla für das Aufdecken von Tippfehlern bis hin zu diversen inhaltlichen Hinweisen, Prof. Dr. Dominik Gruntz für eine Vielzahl an hilfreichen Anmerkungen, Jean-Claude Brantschen und Christian Heitzmann für ihre Kommentare und Tipps und schließlich Maria Herdt für einen kritischen Blick auf einige Kapitel.

Zudem geht ein Dankeschön an das Team des dpunkt.verlags (Dr. Michael Barabas, Dr. Benjamin Ziech, Julia Griebel und Stefanie Weidner) für die tolle Zusammenarbeit. Außerdem möchte ich mich bei Torsten Horn für die fundierte fachliche Durchsicht sowie bei Ursula Zimpfer für ihre Adleraugen beim Copy-Editing bedanken.

Abschließend geht ein lieber Dank an meine Frau Lilija für ihr Verständnis und die Unterstützung. Ihren ganz besonderen Anteil hat unser kleiner Sonnenschein Sophie Jelena dazu beigetragen, indem sie den Papa immer wieder zum Lachen gebracht hat.

Anregungen und Kritik

Trotz großer Sorgfalt und mehrfachen Korrekturlesens lassen sich missverständliche Formulierungen oder teilweise sogar Fehler leider nicht vollständig ausschließen. Falls Ihnen etwas Derartiges auffallen sollte, so zögern Sie bitte nicht, mir dies mitzuteilen. Gerne nehme ich auch Anregungen oder Verbesserungsvorschläge entgegen. Kontaktieren Sie mich bitte per Mail unter:

michael_inden@hotmail.com

Zürich, im Oktober 2023
Michael Inden

2 Schnelleinstieg

2.1 Hallo Welt (Hello World)

Wir beginnen den Einstieg in Java mit einem simplen Programm, nämlich wie traditionell in den allermeisten Büchern zum Programmierenlernen mit HelloWorld, der Ausgabe eines Grußes auf der Konsole. Das haben wir schon in etwas komplizierterer Form in der Einleitung gesehen. Wir wollen es weiter vereinfachen und damit unsere Entdeckungsreise starten.

Java bietet mit der JShell einen interaktiven Kommandozeileninterpreter, auch Read-Eval-Print-Loop (REPL) genannt. Das erleichtert das schrittweise Erlernen von Java, weil wir nicht gleich zu Beginn eine Vielzahl an Konzepten verstehen oder zumindest hinnehmen müssen. Als weiterer Vorteil wird uns eine Menge Tipparbeit erspart.

Die interaktive JShell erlaubt es uns, kleinere Java-Codeschnipsel auszuprobieren. Der Start erfolgt mit dem Kommando `jshell`:

```
$ jshell
|  Willkommen bei JShell - Version 21
|  Geben Sie für eine Einführung Folgendes ein: /help intro

jshell>
```

Neben dem Hinweis auf die Version zeigt der Prompt `jshell>` an, dass wir nun Java-Befehle eingeben können. Probieren wir es gleich einmal aus:

```
jshell> System.out.println("Hello World from JShell!")
Hello World from JShell!
```

Herzlichen Glückwunsch zur ersten Ausgabe mit Java!

Wir verwenden dabei die später in Kapitel 5 vorgestellten Klassen und Methoden. Hier wird über `System.out` eine Ausprägung der Klasse `java.io.PrintStream` referenziert. Diese ermöglicht Ausgaben auf der Konsole. Funktionalität wird durch sogenannte Methoden, eine Folge von Befehlen, bereitgestellt, in diesem Fall durch die Methode mit dem Namen `println()`, der man in runden Klammern einen Text eingerahmt von doppelten Anführungszeichen übergibt. Aber keine Sorge, Sie müssen noch nicht alle Details verstehen – wir werden diese im Verlaufe dieses Buchs ausführlich besprechen. Jetzt geht es zunächst um die ersten Schritte und ein initiales Gespür. Laden Sie die Begleitsourcen herunter oder tippen Sie die einfachen Beispiele ab, und wenn Sie sich sicher fühlen, dann variieren Sie auch gerne ein wenig.

Wir haben bereits einen netten Gruß auf der Konsole ausgeben können. Das war schon ein guter Anfang. Allerdings wäre es etwas eintönig, nur Texte ohne Variation auszugeben. Um den Gruß anzupassen, lernen wir nun Variablen kennen.

2.2 Variablen und Datentypen

Stellen wir uns vor, wir wollten einige Eigenschaften einer Person mit Java verwalten, etwa Name, Alter, Gewicht, Wohnort, Familienstand usw. Dabei helfen uns sogenannte Variablen. Diese dienen zum Speichern und Verwalten von Werten. Dabei gibt es unterschiedliche Typen (auch Datentypen genannt), etwa für Texte, Zahlen und Wahrheitswerte. Wir schauen uns dies zunächst einmal einführend an und erweitern dann unser Wissen Schritt für Schritt, beispielsweise um ergänzende Infos zu Wertebereichen oder Aktionen mit den Variablen. Zum Einstieg betrachten und nutzen wir folgende Typen:

- `String` – Speichert textuelle Informationen, z. B. "Hallo". In Java werden diese in doppelte Anführungszeichen eingeschlossen und Strings genannt.

- `char` – Speichert einzelne Zeichen, wie z. B. 'a' oder 'B'. Solche Zeichenwerte sind von einfachen Anführungszeichen eingeschlossen.
- `int` und `long` – Speichern Ganzzahlen, wie 123 oder -4711. Für `int` geht der Wertebereich von etwa -2 Milliarden bis 2 Milliarden, für `long` ist er noch viel größer.
- `float` und `double` – Speichern Gleitkommazahlen (mit Vor- und Nachkommastellen), wie 72,71 oder -1,357. Achtung: Weil Java dem amerikanischen Sprachraum entstammt, werden als Dezimaltrenner Punkte statt Kommata verwendet. Somit muss es im Java-Programm `72.71` und `-1.357` heißen. Der Typ `float` besitzt eine geringere Genauigkeit und kann somit weniger Nachkommastellen als der Typ `double` verwalten.
- `boolean` – Speichert Wahrheitswerte als wahr oder falsch. In Java sind diese durch `true` oder `false` repräsentiert.

2.2.1 Definition von Variablen

In Java erzeugen wir eine Variable, indem wir zunächst den Typ, dann einen Namen sowie eine Wertzuweisung nach folgendem Muster (Syntax genannt) angeben:

```
typ variablenname = wert;
```

Man spricht in diesem Zusammenhang von der Definition. Eine solche Definition wie auch andere Anweisungen müssen in Java normalerweise mit einem Semikolon beendet werden. In der JShell ist dies oft optional und man kann das Ganze wie folgt schreiben:

```
typ variablenname = wert
```

Lassen Sie uns etwas experimentieren:

```
jshell> int theAnswer = 41
theAnswer ==> 41

jshell> theAnswer = 42
theAnswer ==> 42

jshell> int oneMillion = 1_000_000
oneMillion ==> 1000000

jshell> double PI = 3.1415
PI ==> 3.1415
```

Die JShell bestätigt jede Definition einer Variablen mit einer Ausgabe, wobei zunächst der Variablenname, dann ==> und schließlich der Wert folgt. Für das Beispiel mit einer Million sehen wir die Angabe von Unterstrichen, die sich zur Separierung, hier von Tausendersegmenten, eignet – allerdings nur bei der Angabe, nicht bei der internen Repräsentation.

Den aktuellen Wert einer Variablen können wir durch Eingabe das Namens abfragen bzw. auslesen:

```
jshell> theAnswer
theAnswer ==> 42
```

Wie oben für `theAnswer` angedeutet, kann einer Variablen durchaus mehrmals im Programmverlauf an unterschiedlichen Stellen ein anderer Wert zugewiesen werden. Denken Sie etwa an einen Punktestand in einem Spiel, einen Temperaturverlauf pro Tag und eine Variable `temperature` oder aber an die Modellierung der Tageszeit von morgens, mittags, nachmittags, abends und nachts. Für derart in sich abgeschlossene Wertebereiche existieren Aufzählungen, die wir in Abschnitt 7.4 kennenlernen werden.

Eine solche Variable wird auch bei dem kürzest möglichen Hello-World-Programm automatisch als unbenannte Variable `$1` erzeugt:

```
jshell> "Shortest hello world ever"
$1 ==> "Shortest hello world ever"
```

Besonderer Typ `String`

Wie Zahlen kann man auch Strings mit + verbinden. Man spricht hier von Konkatenation. Als Folge entsteht ein neuer String.

```
jshell> String vorname = "Michael"
vorname ==> "Michael"

jshell> String name = vorname + " " + "Inden"
name ==> "Michael Inden"
```

Wie gezeigt, sind problemlos mehrere Konkatenationen möglich, ebenso eine Verknüpfung von Zahlen mit einem String:

```
jshell> int value = 4711
value ==> 4711

jshell> "Wert: " + value
$3 ==> "Wert: 4711"
```

Der Typ `String` unterscheidet sich von den anderen vorgestellten Typen dadurch, dass er neben + noch einige weitere spezifische Funktionalitäten bereitstellt. Dies werden wir uns genauer in Kapitel 3 anschauen.

Besonderheit beim Typ `char`

Der Typ `char` repräsentiert ein einzelnes Zeichen:

```
jshell> char character = 'A'
character ==> 'A'
```

Interessanterweise lassen sich mathematische Aktionen wie + und - ausführen, allerdings entsteht dann ein Zahlenwert vom Typ `int`. Mithilfe eines später vorgestellten sogenannten ***Casts*** kann man durch die Angabe von `(char)` vor einem Ausdruck diesen wieder in ein Zeichen wandeln (vgl. Abschnitt 7.2.2):

```
jshell> character + 5
$34 ==> 70

jshell> (char)(character + 5)
$35 ==> 'F'
```

Wie gezeigt, kann man sich dann per Addition vorwärts durch das Alphabet bewegen. Eine Subtraktion ist ebenfalls möglich, um eine Anzahl an Buchstaben in Richtung A zu springen.

Kurzschreibweise `var`

Als Kurzschreibweise zur Definition von Variablen gibt es die sogenannte Local Variable Type Inference mit dem reservierten Wort `var`. Was bedeutet das genauer?

Zuvor haben wir zur Definition von Variablen folgende Syntax genutzt:

```
typ variablenname = wert
```

Statt einer konkreten Typangabe können wir kürzer `var` schreiben, um vor allem bei längeren Typangaben etwas Tipparbeit zu sparen.

```
var variablenname = wert
```

Java selbst ist so schlau, festzustellen, was der genaue Typ ist. Insbesondere bei längeren Typnamen wird der Sourcecode durch Nutzung von `var` etwas kürzer und eventuell auch lesbarer:

```
jshell> var theAnswer = 42
theAnswer ==> 42

jshell> var PI = 3.1415;
PI ==> 3.1415

jshell> var name = "Michael"
name ==> "Michael"
```

Eine derart definierte Variable hat dann genau den von Java ermittelten Typ, also ist `theAnswer` vom Typ `int` und `name` vom Typ `String`.

An dem Beispiel möchte ich noch einmal explizit auf zwei Dinge eingehen: Zum einen ist es in der JShell oftmals möglich, auf ein Semikolon am Ende der Anweisungen zu verzichten. Hier wird zur Demonstration nur bei der Definition von `PI` (π)

ein solches verwendet, nicht aber für die beiden anderen Definitionen. Zum anderen kann man zwar `var` statt `int` nutzen, aber dies ist nicht sinnvoll, da es genauso viele Zeichen sind und man somit nichts gewinnt.

Konsolenausgeben – `System.out.println()`

Einleitend hatten wir schon `System.out.println()` genutzt, um Texte auf der Konsole ausgeben zu können. Das lässt sich auch mit Variablen kombinieren:

```
jshell> var name = "Michael"
name ==> "Michael"

jshell> System.out.println("Hello " + name)
Hello Michael
```

Auf ein Detail möchte ich nochmals hinweisen: Bei `println()` handelt es sich um eine in Java integrierte sogenannte Methode. Diese dienen dazu, gewisse Aktionen standardisiert bereitzustellen. Methoden schauen wir uns in Abschnitt 2.5 genauer an.

Besonderheit Deklaration

Manchmal weiß man zwar bereits, dass man das Ergebnis einer Berechnung in einer Variablen speichern möchte, hat diese Berechnung aber noch nicht ausgeführt. Für solche Fälle ist es möglich, Variablen ohne Wertzuweisung zu notieren, man spricht dabei von Deklarieren. Die Wertzuweisung erfolgt somit nicht direkt, sondern erst später im Ablauf – hier durch eine Konsolenausgabe als zwischenzeitliche Aktion angedeutet:

```
jshell> int age
age ==> 0

jshell> System.out.println("Something in between!")
Something in between!

jshell> age = 15
age ==> 15
```

Oftmals ist die bereits besprochene Definition, also die Kombination aus Deklaration und direkter Wertzuweisung, zu bevorzugen, da sich dies klarer liest und Fehlverwendungen vorbeugt. Außerdem kann man bei einer Deklaration `var` nicht verwenden, weil kein Wert angegeben wird und somit der Typ nicht ermittelt werden kann.

Konstanten – finale Variablen

Mithilfe des Schlüsselworts `final` vor der Definition einer Variablen machen wir diese unveränderlich und erzeugen somit eine Konstante:

```
final double PI = 3.1415
PI = 3.14;  // error: cannot assign a value to a final variable
```

Keine Konstanten in der JShell

Bitte beachten Sie, dass die JShell `final` nicht so unterstützt wie ein Java-Programm. Zwar kann man `final` angeben, aber es entsteht keine Konstante, sondern eine Variable. Dadurch ist eine zweite Zuweisung in der JShell möglich und verändert einfach den Wert:

```
jshell> final double PI = 3.1415
PI ==> 3.1415

jshell> PI = 3
PI ==> 3.0
```

2.2.2 Bezeichner (Variablennamen)

Ohne es explizit zu erwähnen, haben wir uns in den Beispielen bei der Benennung von Variablen an ein paar Regeln gehalten. Zunächst einmal muss jede Variable (und auch die später vorgestellten Methoden und Klassen) durch einen eindeutigen Namen, auch Identifier oder Bezeichner genannt, gekennzeichnet werden.

Bei der Benennung sollte man Folgendes beachten:

- Namen sollten mit einem Buchstaben beginnen[1] – Ziffern sind als erstes Zeichen eines Variablennamens nicht erlaubt.
- Namen können danach aus einem beliebigen Mix aus Buchstaben (auch Umlauten), Ziffern, $ und _ bestehen, dürfen aber keine Leerzeichen enthalten.
- Bestehen Namen aus mehreren Wörtern, dann ist es guter Stil, die sogenannte CamelCase-Schreibweise zu verwenden. Bei dieser beginnt das erste Wort mit einem Kleinbuchstaben und danach startet jedes neue Wort mit einem Großbuchstaben, etwa `arrivalTime`, `estimatedDuration`, `shortDescription`.
- Die Groß- und Kleinschreibung von Namen spielt eine Rolle: `arrivalTime` und `arrivaltime` bezeichnen unterschiedliche Variablen.
- Namen dürfen nicht mit den in Java vordefinierten und in Anhang A aufgelisteten Schlüsselwörtern übereinstimmen, demzufolge sind z. B. `public`, `class` oder `int` keine gültigen Namen für Variablen. Die Begriffe `PublicTransport`, `classic` oder `Winter` hingegen sind erlaubt.

Übrigens spielt es für Java keine Rolle, ob Sie sehr kurze (`i`, `m`, `p`), kryptische (`dpy`, `mtr`) oder aber gut gewählte Namen (`daysPerYear`, `maxTableRows`) nutzen. Für Sie und das Verständnis beim Nachvollziehen eines Java-Programms macht das aber einen himmelweiten Unterschied. Natürlich sind kurze Variablennamen wie etwa `x` und `y` zur Bezeichnung von Koordinaten vollkommen verständlich, aber was sagt beispielsweise ein einzelner Buchstabe `m` aus?

[1] Namen dürfen auch mit $ oder _ beginnen, jedoch ist das eher für fortgeschrittene Themen nützlich. Da wir in diesem Buch auf Verständlichkeit setzen, werden wir das nicht nutzen.

Betrachten wir zur Verdeutlichung ein Beispiel:

```
// Gut verständliche Namen
int minutesPerHour = 60
int daysPerYear = 365
int maxTableRows = 25

// NAJA, oft schwieriger verständlich, was die Abkürzungen bedeuten
int m = 60;
int dpy = 365
int mtr = 25

// ACHTUNG: ziemlich schwierig unterscheidbar
String arrivalTime = "16:50"
double arrivaltime = 16.50
```

Die letzten beiden Variablendefinitionen sind ungünstig, da sie kaum zu unterscheiden sind. Besäßen diese denselben Typ, so käme es leicht zu Fehlverwendungen.

Im Listing sehen wir Kommentare. Damit kann man erklärende Zusatzinformationen hinterlegen. In diesem Fall werden mit // einzeilige Kommentare eingeleitet. Alles, was nach dem // bis zum Ende der Zeile folgt, wird bei der Ausführung von Java ignoriert. Weitere Varianten von Kommentaren lernen wir in Abschnitt 2.7 kennen.

2.3 Operatoren im Überblick

Operatoren dienen dazu, Operationen zwischen Variablen und / oder Werten auszuführen. Selbst eine banale Addition ist ein Beispiel, nämlich für den Operator +:

```
jshell> int sum1 = 200 + 50;
sum1 ==> 250

jshell> int sum2 = sum1 + 500;
sum2 ==> 750

jshell> int sum3 = sum2 + 750;
sum3 ==> 1500
```

Inhaltsverzeichnis

O'REILLY®

Java lernen – kurz & gut

Dieses Buch ist für vielbeschäftigte Programmierer:innen, die eine knappe und dennoch gut verständliche Einführung in Java als eine seit Jahren populäre Programmiersprache suchen. *Java lernen – kurz & gut* bietet einen unterhaltsamen Einstieg und informiert Sie über viele Java-Themen, die Ihnen helfen werden, schnell durchzustarten:

- **Installation von Java und einer Entwicklungsumgebung**
- **Schnelleinstieg in die wichtigsten Aspekte**
- **Basisbausteine wie Strings, Arrays, Zufallszahlen, Fallunterscheidungen und Schleifen**
- **Klassen und objektorientierte Programmierung**
- **Datencontainer wie Listen, Mengen und Maps**
- **Fortgeschrittene Themen zu Collections wie Lambdas und Streams**
- **Datumsverarbeitung inklusive Berechnungen**
- **Dateiverarbeitung und Behandlung von Fehlern mit Exceptions**

Trotz seines kompakten Formats liefert dieses Buch eine fundierte Einführung und eine Fülle an leicht nachvollziehbaren Beispielen, die zum Experimentieren einladen. Es unterstützt Sie optimal dabei, Ihre Java-Kenntnisse auf- und auszubauen. Insbesondere wenn Sie bereits ein wenig mit z.B. C++ oder C# vertraut sind, ist dieses Buch die ideale Wahl, um fundiert in Java einzusteigen und eigene Experimente zu beginnen.

www.oreilly.de

9 783960 092049

plus+ Interesse am E-Book?
www.oreilly.plus

Euro 16,90 (D)
ISBN 978-3-96009-204-9