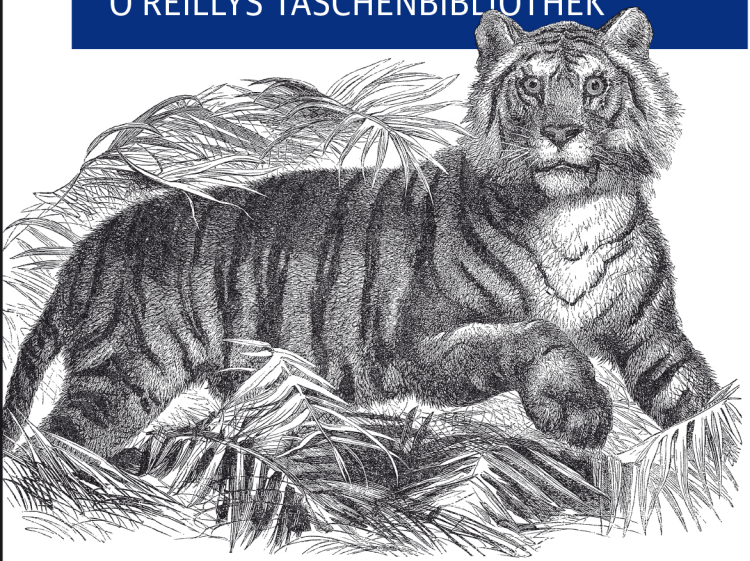
O'REILLY®

3. Auflage
Behandelt Java 8 & 9

Java
kurz & gut

O'REILLYS TASCHENBIBLIOTHEK



Robert Liguori & Patricia Liguori

Übersetzung von
Lars Schulten & Thomas Demmig

3. AUFLAGE

Java

kurz & gut

Robert Liguori & Patricia Liguori

O'REILLY®

Robert Liguori, Patricia Liguori

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Satz: III-Satz, www.drei-satz.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Druck und Bindung: Media-Print Informationstechnologie,
www.mediaprint-druckerei.de

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.

ISBN:
Print 978-3-96009-051-9
PDF 978-3-96010-210-6
ePub 978-3-96010-211-3
mobi 978-3-96010-212-0

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

3. Auflage

Wieblinger Weg 17
69123 Heidelberg

Authorized German translation of the English edition of Java Pocket Guide, 4th Edition, ISBN 9781491938690 © 2017 Gliesian, LLC. This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

5 4 3 2 1 0

Dieses Buch ist unserer wunderbaren Tochter Ashleigh gewidmet.

Inhalt

Teil II: Plattform

Teil III: Anhänge

Vorwort

Diese Taschenreferenz soll Ihnen ein ständiger Begleiter sein und Ihnen als Kurzreferenz für die Grundeinrichtung der Programmiersprache Java sowie der Java-Plattform dienen.

Java kurz & gut liefert Ihnen die Informationen, die Sie benötigen, wenn Sie Java-Programme entwickeln oder debuggen. Das schließt nützliche Programmbeispiele, Tabellen, Abbildungen und Listen ein.

Das Buch behandelt Java bis Java SE 9 einschließlich eines Teils der über 80 *JDK Enhancement Proposals* (JEPs). Zudem geht es auf die neue Java-Shell und das neue Java Module System ein.

Die Codebeispiele in diesem Buch entstammen größtenteils der Gliesians Web Application (*https://gliesians.com/index-genealogy.faces*). Aktuell liegt deren Hauptfokus auf dem Bereitstellen kostenloser Tools rund um die Genealogie und kleiner Drohnen.

Der Stoff in diesem Buch unterstützt Sie außerdem bei der Vorbereitung auf die Prüfung zum Oracle Certified Programmer.

Der Aufbau dieses Buchs

Dieses Buch hat die drei Teile: Teil I, »Sprache«, beschreibt die Programmiersprache Java in der Form, in der sie von der *Java Language Specification* (JLS) und in den JEPs formuliert wird. Teil II, »Plattform«, geht auf Bestandteile der Java-Plattform und darauf bezogene Themen ein. Die Anhänge in Teil III erläutern zusätzliche Technologien.

Typografische Konventionen

In diesem Buch werden die folgenden typografischen Konventionen verwendet:

Kursiv
: Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateinamenserweiterungen.

`Nicht-Proportional-Schrift`
: Wird für Programmcode verwendet und außerdem in Textabsätzen, um Programmelemente wie Variablen oder Funktionsnamen, Datenbanken, Umgebungsvariablen, Anweisungen und Schlüsselwörter zu kennzeichnen.

`Nicht-Proportional-Schrift fett`
: Kennzeichnet Befehle oder anderen Text, den der Benutzer wörtlich eingeben muss.

`Nicht-Proportional-Schrift kursiv`
: Kennzeichnet Text, der durch vom Benutzer vorgegebene oder durch den Kontext bestimmte Werte ersetzt werden muss.

Dieses Element kennzeichnet einen Tipp, einen Vorschlag oder eine allgemeine Anmerkung.

Danksagungen

Unser besonderer Dank gilt all den Leuten bei O'Reilly. Sehr unterstützt haben uns auch Greg Grockenberger und Ryan Cuprak, die zu den Kapiteln über JShell und das Java Module System beigetragen haben. Ryan war zudem an der technischen Begutachtung des Buchs beteiligt.

Außerdem möchten wie allen danken, die am ursprünglichen *Java Pocket Guide*, dem *Java 7 Pocket Guide* und dem *Java 8 Pocket Guide* mitgewirkt haben.

Ein weiterer Dank geht an Personen, die nichts mit diesem Buchprojekt zu tun haben: Don Anderson, David Chong, Keith Cianfrani, Jay Clark, Steve Cullen, Ed DiCampli, Phil Greco, Scott Houck, Cliff Johnson, Juan Keller, Fran Kelly, Mike Krauss, Mike Lazlo, Phil Maloney, Lana Manovych, Matt Mariani, Chris Martino, Roe Morande, Sohrob Mottaghi, Brendan Nugent, Keith Smaniotto, Tom Tessitore, Lacey Thompson, Tyler Travis, Justin Trulear und Jack Wombough.

TEIL I

Sprache

KAPITEL 1

Namenskonventionen

Namenskonventionen dienen dazu, Java-Programme lesbarer zu machen. Es ist wichtig, dass man aussagekräftige und eindeutige Namen wählt, die aus Java-Buchstabenzeichen bestehen. Die folgenden Beispiele stammen aus verschiedensten Java-Quellen.

Akronyme

Wenn Sie in einem Namen ein Akronym verwenden, sollte nur der erste Buchstabe ein Großbuchstabe sein – und auch nur dann, wenn dort ein Großbuchstabe angemessen ist:

```
// zum Beispiel wird DNA als Dna genutzt
public class GliesianDnaProvider {...}

// Most Recent Common Ancestor (MRCA) ist Mrca
public class MrcaCalculator {...}
```

Annotationsnamen

Annotationsnamen werden in der Java SE-API bei den vordefinierten Annotationstypen auf unterschiedliche Weise formuliert, [Adjektiv|Verb][Nomen]:

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface FunctionalInterface {}
```

Klassennamen

Klassennamen sollten Nomen sein, da Klassen »Dinge« oder »Objekte« repräsentieren. Sie sollten gemischte Groß-/Kleinschreibung (CamelCase) verwenden, wobei nur der erste Buchstabe jedes Worts großgeschrieben wird, z. B.:

```
public class AirDensityCalculator {...}
```

Namen von Konstanten

Namen von Konstanten sollten gänzlich aus Großbuchstaben bestehen, und mehrere Wörter sollten durch Unterstriche getrennt werden:

```
private static final double KELVIN = 273.16;
private static final double DRY_AIR_GAS_CONSTANT = 287.058;
private static final double HUMID_AIR_GAS_CONSTANT = 461.4964;
```

Enumerationsnamen

Enumerationsnamen verwenden die gleichen Namenskonventionen wie Klassennamen. Die Namen der Elemente der Enumeration (die Optionen) sollten gänzlich aus Großbuchstaben bestehen:

```
public enum MeasurementSystem {
  METRIC, UNITED_STATES_CUSTOMARY, IMPERIAL
}

public enum Study {
  ALL, NON_ENDOGAMOUS, SEMI_ENDOGAMOUS, ENDOGAMOUS
}

public enum RelationshipMatchCategory {
  IMMEDIATE, CLOSE, DISTANT, SPECULATIVE
}
```

Die Namen generischer Typparameter

Die Namen generischer Typparameter sollten aus nur einem Großbuchstaben bestehen. Üblicherweise wird der Buchstabe `T` (für Typ) verwendet.

Das Collections-Framework macht umfassenden Gebrauch von Generics. `E` wird für Collection-Elemente genutzt, `S` für Service-Loader sowie `K` und `V` für Schlüssel (Key) und Werte (Value) in Maps:

```
public interface Map <K,V> {
  V put(K key, V value);
}
```

Namen von Instanzvariablen und statischen Variablen

Die Namen von Instanzvariablen und statischen Variablen sollten Nomen sein und den gleichen Namenskonventionen folgen wie Methodennamen:

```
private String prediction;
```

Interface-Namen

Interface-Namen sollten Adjektive sein, die auf »able« oder »ible« enden, wenn ein Interface eine Fähigkeit darstellt; andernfalls sollten Interface-Namen ebenfalls Nomen sein. Interface-Namen folgen den gleichen Konventionen für die Groß-/Kleinschreibung wie Klassennamen:

```
public interface Relatable {...}
public interface SystemPanel {...}
```

Methodennamen

Methodennamen sollten ein Verb enthalten, da Methoden genutzt werden, um Objekte etwas unternehmen zu lassen. Die Groß-/Kleinschreibung sollte erneut gemischt sein, wobei der Name hier idealerweise mit einem Kleinbuchstaben beginnt und die ersten Buchstaben aller nachfolgenden Wörter großgeschrieben werden sollten. Methodennamen können Adjektive und Nomen enthalten:

```
public void clear() {...} // Verb
public String toString() {...} // Präposition und Nomen
public double getDryAirDensity() {...} // Verb, Adjektiv und
Nomen
public double getHumidAirDensity() {...} // Verb, Adjektiv und
Nomen
```

Package-Namen

Package-Namen sollten einzigartig sein und vollständig aus Kleinbuchstaben bestehen. Bei Bedarf können Unterstriche gesetzt werden:

```
// Gliesian.com (Firma), JAirDensity (Software)
package com.gliesian.jairdensity;

// Gliesian.com (Firma), FOREX Calculator (Software), Utilities
package com.gliesian.forex_calculator.utils;
```

Der Name öffentlich zugänglicher Packages sollte der umgekehrte Domainname der Organisation sein, der mit der aus einem Wort bestehenden Top-Level-Domain (d. h. *com*, *net*, *org* oder *edu*) beginnt, auf die der Name der Organisation und der Name des Projekts oder der Abteilung folgen. (Interne Packages werden üblicherweise nach dem Projekt benannt.)

Package-Namen, die mit `java` oder `javax` beginnen, sind reserviert und dürfen nur für Implementierungen verwendet werden, die den Java-Klassenbibliotheken entsprechen.

Modulnamen

Modulnamen sollten wie Paketnamen aus umgekehrten Domainnamen bestehen (mit den gleichen Richtlinien):

```
module com.gliesian.utils {
}
```

Namen von Parametern und lokalen Variablen

Die Namen von Parametern oder lokalen Variablen sollten aussagekräftige (kleingeschriebene) einzelne Wörter, Akronyme oder Abkürzungen sein. Wenn mehrere Wörter erforderlich sind, folgt die Groß-/Kleinschreibung der Namen den gleichen Konventionen wie die der Methodennamen:

```
public void printPredictions(ArrayList predictions) {
  int counter = 1;
  for (String prediction : predictions) {
    System.out.println("Vorhersage Nr. " + counter++ +
      ": " + prediction);
  }
}
```

Die Namen temporärer Variablen können auch aus nur einem einzigen Buchstaben bestehen, z. B. `i`, `j`, `k`, `m` und `n` für ganze Zahlen sowie `c`, `d` und `e` für Zeichen. Für temporäre und Schleifenvariablen werden einbuchstabige Namen wie in Tabelle 1-1 vorgeschlagen.

Tabelle 1-1: Temporäre und Schleifenvariablen

Einbuchstabiger Name	Typ
`b`	Byte
`c`	Character
`d`	Double
`e`	Exception
`f`	Float
`i`, `j` oder `k`	Integer
`l`	Long
`o`	Object
`s`	String

KAPITEL 2

Lexikalische Elemente

Java-Quellcode besteht aus Wörtern oder Symbolen, die als *lexikalische Elemente* oder *Token* bezeichnet werden. Zu den lexikalischen Elementen von Java zählen Zeilenendezeichen, Leerraumzeichen, Kommentare, Schlüsselwörter, Bezeichner, Trennzeichen, Operatoren und Literale. Die Wörter und Symbole bestehen in der Programmiersprache Java aus Zeichen aus dem Unicode-Zeichensatz.

Unicode und ASCII

Der von der Standardisierungsorganisation Unicode Consortium gepflegte Unicode ist ein allgemeiner Zeichensatz, dessen erste 128 Zeichen denen des Zeichensatzes ASCII (*American Standard Code for Information Interchange*) entsprechen. Unicode stellt eine eindeutige Nummer für jedes Zeichen bereit und kann auf allen Plattformen, in allen Programmen und mit allen Sprachen verwendet werden. Java SE 9 nutzt Unicode 8.0.0. Mehr Informationen dazu finden Sie im Onlinehandbuch (*http://www.unicode.org/versions/Unicode8.0.0/*). Java SE 8 nutzt Unicode 6.2.0.

Kommentare, Bezeichner und Stringliterale sind in Java nicht auf ASCII-Zeichen beschränkt. Alle anderen Java-Eingabeelemente werden aus ASCII-Zeichen gebildet.

Die Unicode-Version, die von der jeweiligen Java-Plattform verwendet wird, wird in der `Character`-Klasse der Java-API dokumentiert. Auf eine Liste der Unicode-Zeichencodes für Schriftzeichen,

Symbole und Interpunktionszeichen können Sie unter *http://unicode.org/charts/* zugreifen.

Druckbare ASCII-Zeichen

ASCII reserviert Code 32 (Leerzeichen) und die Codes 33 bis 126 (Buchstaben, Ziffern, Interpunktionszeichen und ein paar weitere Zeichen) für druckbare Zeichen. Tabelle 2-1 zeigt die dezimalen Werte und die ASCII-Zeichen, die diesen Codes entsprechen.

Tabelle 2-1: Druckbare ASCII-Zeichen

32 SP	48 0	64 @	80 P	96 '	112 p
33 !	49 1	65 A	81 Q	97 a	113 q
34 "	50 2	66 B	82 R	98 b	114 r
35 #	51 3	67 C	83 S	99 C	115 S
36 $	52 4	68 D	84 T	100 d	116 t
37 %	53 5	69 E	85 U	101 e	117 u
38 &	54 6	70 F	86 V	102 f	118 v
39 '	55 7	71 G	87 W	103 g	119 w
40 (	56 8	72 H	88 X	104 h	120 x
41)	57 9	73 I	89 Y	105 i	121 y
42 *	58 :	74 J	90 Z	106 j	122 z
43 +	59 ;	75 K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45 -	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	

Nicht druckbare ASCII-Zeichen

ASCII reserviert die Codes 0 bis 31 sowie Code 127 für *Steuerzeichen*. Tabelle 2-2 zeigt die dezimalen Werte und die ASCII-Zeichen für den jeweiligen Code.

Tabelle 2-2: Nicht druckbare ASCII-Zeichen

00 NUL	07 BEL	14 SO	21 NAK	28 FS
01 SOH	08 BS	15 SI	22 SYN	29 GS
02 STX	09 HT	16 DLE	23 ETB	30 RS
03 ETX	10 NL	17 DC1	24 CAN	31 US
04 EOT	11 VT	18 DC2	25 EM	127 DEL
05 ENQ	12 NP	19 DC3	26 SUB	
06 ACK	13 CR	20 DC4	27 ESC	

ASCII 10 ist ein Zeilenvorschub, ASCII 13 ein Wagenrücklauf.

Kompakte Strings

Kompakte Strings sind eine Optimierung, die es erlaubt, Strings speichereffizienter abzulegen. In Java 9 ist sie standardmäßig aktiviert. Um sie abzuschalten, nutzen Sie `-XX:-CompactStrings+`, wenn Sie vor allem UTF16-Strings nutzen.

Kommentare

Ein einzeiliger Kommentar beginnt mit zwei Schrägstrichen und endet unmittelbar vor dem Zeilenendezeichen:

```
// Standard-Geburtsjahr eines Kindes
private Integer childsBIrthYear = 1950;
```

Ein mehrzeiliger Kommentar beginnt mit einem Schrägstrich, auf den unmittelbar ein Asterisk folgt, und endet mit einem Asterisk, auf den unmittelbar ein Schrägstrich folgt. Ein einzelner Asterisk zu Anfang der einzelnen Kommentarzeilen ist eine nützliche Formatierungskonvention; das ist üblich, aber nicht erforderlich:

```
/*
 * Das Durchschnittsalter einer Frau bei der Geburt
 * ihres ersten Kindes betrug in den USA im Jahr 2001
 * 24.9 Jahre. Daher werden wir den Wert von 25 Jahren
 * als Standardvorschlag nutzen.
 */
private Integer mothersAgeGivingBirth = 25;
```

Ein Javadoc-Kommentar wird vom Javadoc-Werkzeug verarbeitet, um eine API-Dokumentation im HTML-Format zu generieren. Ein Javadoc-Kommentar beginnt mit einem Schrägstrich, auf den unmittelbar zwei Asteriske folgen, und schließt mit einem Asterisk, auf den unmittelbar ein Schrägstrich folgt (*Oracles Dokumentationsseite* (*http://bit.ly/16mhGeT*) bietet weitere Informationen zum Javadoc-Werkzeug):

```
/**
 * Genitor Birthdate Predictor
 *
 * @author Robert J. Liguori
 * @author Gliesian, LLC.
 * @version 0.1.1 09-02-16
 * @since 0.1.0 09-01-16
 */
public class GenitorBirthdatePredictorBean {...}
```

Kommentare dürfen in Java nicht geschachtelt werden:

```
/* Das hier ist /* in Java */ nicht gestattet. */
```

Schlüsselwörter

Tabelle 2-3 enthält die Schlüsselwörter von Java 9. Zwei davon, `const` und `goto`, sind reserviert, werden von Java aber nicht verwendet.

Java-Schlüsselwörter dürfen in Java-Programmen nicht als Bezeichner verwendet werden.