Michael Inden

Einfach Java

Gleich richtig programmieren lernen

dpunkt.verlag

Dipl.-Inform. Michael Inden ist Oracle-zertifizierter Java-Entwickler. Nach seinem Studium in Oldenburg hat er bei diversen internationalen Firmen in verschiedenen Rollen etwa als Softwareentwickler, -architekt, Consultant, Teamleiter, CTO sowie Leiter Academy gearbeitet. Zurzeit ist er freiberuflich als Autor und Trainer in Zürich tätig.

Michael Inden hat über zwanzig Jahre Berufserfahrung beim Entwurf komplexer Softwaresysteme gesammelt, an diversen Fortbildungen und mehreren Java-One-Konferenzen teilgenommen. Sein besonderes Interesse gilt dem Design qualitativ hochwertiger Applikationen sowie dem Coaching. Sein Wissen gibt er gerne als Trainer in internen und externen Schulungen und auf Konferenzen weiter, etwa bei der JAX/W-JAX, JAX London, Oracle Code One, ch.open sowie bei der Java User Group Switzerland.

Michael Inden

Einfach Java

Gleich richtig programmieren lernen

dpunkt.verlag

Michael Inden
michael_inden@hotmail.com

Lektorat: Michael Barabas
Projektkoordinierung: Anja Weimer
Fachgutachten: Torsten Horn, Aachen
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Michael Inden
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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 978-3-86490-852-1
PDF 978-3-96910-543-6
ePub 978-3-96910-544-3
mobi 978-3-96910-545-0

1. Auflage 2021

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

5 4 3 2 1 0

Für unsere bezaubernde Prinzessin Sophie Jelena

Inhaltsverzeichnis

Vorwort ..... **xv**

I Einstieg 1

1 Einführung **3**
1.1 Java im Überblick 3
1.2 Los geht's – Installation 5
1.2.1 Java-Download 5
1.2.2 Installation des JDKs 7
1.2.3 Nacharbeiten nach der Java-Installation 7
1.2.4 Java-Installation prüfen 8
1.3 Entwicklungsumgebungen 9
1.3.1 Installation von Eclipse 10
1.3.2 Eclipse starten 12
1.3.3 Erstes Projekt in Eclipse 14
1.3.4 Erste Klasse in Eclipse 16

2 Schnelleinstieg **19**
2.1 Hallo Welt (Hello World) 19
2.2 Variablen und Datentypen 20
2.2.1 Definition von Variablen 20
2.2.2 Bezeichner (Variablennamen) 23
2.3 Operatoren im Überblick 25
2.3.1 Arithmetische Operatoren 25
2.3.2 Zuweisungsoperatoren 28
2.3.3 Vergleichsoperatoren 30
2.3.4 Logische Operatoren 31
2.4 Fallunterscheidungen 32

2.5 Methoden 36
2.5.1 Methoden aus dem JDK nutzen 36
2.5.2 Eigene Methoden definieren 37
2.5.3 Nützliche Beispiele aus dem JDK 40
2.5.4 Signatur einer Methode 43
2.5.5 Fehlerbehandlung und Exceptions 44
2.6 Kommentare 44
2.7 Schleifen 45
2.7.1 Die `for`-Schleife 45
2.7.2 Die for-each-Schleife 46
2.7.3 Die `while`-Schleife 47
2.7.4 Die `do-while`-Schleife 48
2.8 Rekapitulation 49
2.9 Weiterführende Dokumentation für nächste Schritte 50
2.10 Aufgaben und Lösungen 51
2.10.1 Aufgabe 1: Mathematische Berechnungen 51
2.10.2 Aufgabe 2: Methode und `if` 51
2.10.3 Aufgabe 3: Selbstabholerrabatt 52
2.10.4 Aufgabe 4: Schleifen mit Berechnungen 53
2.10.5 Aufgabe 5: Schleifen und fixe Schrittweite 54
2.10.6 Aufgabe 6: Schleifen mit variabler Schrittweite 54
2.10.7 Aufgabe 7: Verschachtelte Schleifen – Variante 1 55
2.10.8 Aufgabe 8: Verschachtelte Schleifen – Variante 2 56
2.10.9 Aufgabe 9: Verschachtelte Schleifen – Variante 3 57

3 Strings 59
3.1 Schnelleinstieg 59
3.1.1 Gebräuchliche Stringaktionen 59
3.1.2 Suchen und Ersetzen 63
3.1.3 Informationen extrahieren und formatieren 65
3.2 Nächste Schritte 67
3.2.1 Die Klasse `Scanner` 68
3.2.2 Mehrzeilige Strings (Text Blocks) 71
3.2.3 Strings und `char[]`s 72
3.3 Praxisbeispiel: Text in Title Case wandeln 74
3.4 Aufgaben und Lösungen 76
3.4.1 Aufgabe 1: Länge, Zeichen und Enthaltensein 76
3.4.2 Aufgabe 2: Title Case mit `Scanner` 76
3.4.3 Aufgabe 3: Zeichen wiederholen 77
3.4.4 Aufgabe 4: Vokale raten 78
3.4.5 Aufgabe 5: String Merge 80

4 Arrays ... 83
4.1 Schnelleinstieg ... 83
4.1.1 Gebräuchliche Aktionen ... 83
4.1.2 Mehrdimensionale Arrays ... 89
4.2 Nächste Schritte ... 90
4.2.1 Eindimensionale Arrays ... 91
4.2.2 Mehrdimensionale Arrays ... 95
4.3 Praxisbeispiel: Flächen füllen ... 99
4.4 Aufgaben und Lösungen ... 102
4.4.1 Aufgabe 1: Durcheinanderwürfeln eines Arrays ... 102
4.4.2 Aufgabe 2: Arrays kombinieren ... 103
4.4.3 Aufgabe 3: Rotation um eine oder mehrere Positionen ... 104
4.4.4 Aufgabe 4: Zweidimensionales String-Array ausgeben ... 106
4.4.5 Aufgabe 5: Dreieckiges Array: Upside Down ... 107

5 Klassen und Objektorientierung ... 109
5.1 Schnelleinstieg ... 109
5.1.1 Grundlagen zu Klassen und Objekten ... 110
5.1.2 Eigenschaften (Attribute) ... 112
5.1.3 Verhalten (Methoden) ... 117
5.1.4 Objekte vergleichen – die Rolle von `equals()` ... 120
5.2 Nächste Schritte ... 123
5.2.1 Klassen ausführbar machen ... 123
5.2.2 Imports und Packages ... 126
5.2.3 Übergang zum Einsatz einer IDE ... 128
5.2.4 Imports und Packages: Auswirkungen auf unsere Applikation 132
5.2.5 Verstecken von Informationen ... 138
5.3 Vererbung ... 142
5.3.1 Basisklassen und abstrakte Basisklassen ... 143
5.3.2 Overloading und Overriding ... 145
5.4 Die Klasse `Object` ... 147
5.4.1 Beispielklasse `Person` ... 148
5.4.2 Die Methode `toString()` ... 149
5.4.3 Ergänzungen zur Methode `equals(Object)` ... 150
5.4.4 Typprüfung mit `instanceof` ... 150
5.4.5 Pattern Matching bei `instanceof` ... 151
5.5 Schnittstelle (Interface) und Implementierung ... 152
5.6 Records ... 154
5.7 Aufgaben und Lösungen ... 156
5.7.1 Aufgabe 1: Obstkorb ... 156
5.7.2 Aufgabe 2: Superheld ... 157
5.7.3 Aufgabe 3: Zähler ... 159
5.7.4 Aufgabe 4: Zähler mit Überlauf ... 161

6 Collections ... 165
6.1 Schnelleinstieg ... 165
6.1.1 Die Klasse `ArrayList` ... 165
6.1.2 Die Klasse `HashSet` ... 171
6.1.3 Iteratoren ... 175
6.1.4 Die Klasse `HashMap` ... 177
6.2 Nächste Schritte ... 181
6.2.1 Generische Typen (Generics) ... 181
6.2.2 Basisinterfaces für die Containerklassen ... 186
6.3 Praxisbeispiel: Einen Stack selbst realisieren ... 190
6.4 Aufgaben und Lösungen ... 192
6.4.1 Aufgabe 1: Tennisverein-Mitgliederliste ... 192
6.4.2 Aufgabe 2: Liste mit Farbnamen füllen und filtern ... 193
6.4.3 Aufgabe 3: Duplikate entfernen – Variante 1 ... 193
6.4.4 Aufgabe 4: Duplikate entfernen – Variante 2 ... 194
6.4.5 Aufgabe 5: Hauptstädte ... 195
6.4.6 Aufgabe 6: Häufigkeiten von Namen ... 195
6.4.7 Aufgabe 7: Objekte mit Maps selbst gebaut ... 196
6.4.8 Aufgabe 8: Listenreihenfolge umdrehen (mit Stack) ... 197

7 Ergänzendes Wissen ... 199
7.1 Sichtbarkeits- und Gültigkeitsbereiche ... 199
7.2 Primitive Typen und Wrapper-Klassen ... 201
7.2.1 Grundlagen ... 201
7.2.2 Casting: Typerweiterungen sowie -verkleinerungen ... 206
7.2.3 Konvertierung von Werten ... 207
7.3 Ternary-Operator (?-Operator) ... 210
7.4 Aufzählungen mit `enum` ... 211
7.5 Switch ... 213
7.6 Moderne Switch Expressions ... 215
7.6.1 Einführendes Beispiel ... 215
7.6.2 Weitere Gründe für die Neuerung ... 217
7.7 Pattern Matching bei Switch Expressions (Java 17 Preview) ... 219
7.7.1 Einführendes Beispiel ... 219
7.7.2 Spezialitäten ... 220
7.8 `Break` und `Continue` in Schleifen ... 222
7.8.1 Funktionsweise von `break` und `continue` in Schleifen ... 222
7.8.2 Wie macht man es besser? ... 224
7.9 Rekursion ... 226

7.10 Aufgaben und Lösungen ... 229
7.10.1 Aufgabe 1: Würfelspiel ... 229
7.10.2 Aufgabe 2: Prüfung auf Vokale mit `switch` ... 230
7.10.3 Aufgabe 3: Temperaturumrechnung ... 230
7.10.4 Aufgabe 4: Palindrom-Prüfung mit Rekursion ... 232

II Aufstieg 235

8 Mehr zu Klassen und Objektorientierung ... 237
8.1 Wissenswertes zu Vererbung ... 237
8.1.1 Generalisierung und Spezialisierung ... 237
8.1.2 Polymorphie ... 239
8.1.3 Sub-Classing und Sub-Typing ... 240
8.2 Varianten innerer Klassen ... 241
8.2.1 »Normale« innere Klassen ... 241
8.2.2 Statische innere Klassen ... 242
8.2.3 Methodenlokale innere Klassen ... 243
8.2.4 Anonyme innere Klassen ... 243

9 Lambdas und Streams ... 245
9.1 Einstieg in Lambdas ... 245
9.1.1 Syntax von Lambdas ... 245
9.1.2 Functional Interfaces und SAM-Typen ... 246
9.2 Methodenreferenzen ... 249
9.3 Externe vs. interne Iteration ... 250
9.3.1 Externe Iteration ... 251
9.3.2 Interne Iteration ... 251
9.3.3 Das Interface `Predicate<T>` ... 252
9.4 Streams im Überblick ... 253
9.4.1 Streams erzeugen – Create Operations ... 254
9.4.2 Intermediate und Terminal Operations im Überblick ... 255
9.4.3 Zustandslose Intermediate Operations ... 255
9.4.4 Zustandsbehaftete Intermediate Operations ... 260
9.4.5 Terminal Operations ... 261
9.5 Aufgaben und Lösungen ... 264
9.5.1 Aufgabe 1: Erwachsene aus Personenliste extrahieren ... 264
9.5.2 Aufgabe 2: Stream-API ... 265
9.5.3 Aufgabe 3: Informationen mit Stream-API extrahieren ... 266
9.5.4 Aufgabe 4: Häufigkeiten von Namen ... 267
9.5.5 Aufgabe 5: Kollektoren ... 267

10 Verarbeitung von Dateien ... 269
10.1 Schnelleinstieg ... 269
10.1.1 Das Interface `Path` und die Utility-Klasse `Files` ... 270
10.1.2 Anlegen von Dateien und Verzeichnissen ... 270
10.1.3 Inhalt eines Verzeichnisses auflisten ... 271
10.1.4 Pfad ist Datei oder Verzeichnis? ... 272
10.1.5 Dateiaktionen und die Utility-Klasse `Files` ... 272
10.1.6 Informationen zu `Path`-Objekten ermitteln ... 276
10.1.7 Kopieren ... 277
10.1.8 Umbenennen ... 279
10.1.9 Löschen ... 280
10.2 Dateibehandlung und die Klasse `File` ... 281
10.2.1 Konvertierung von `Path` in `File` und zurück ... 282
10.2.2 Die Klasse `File` im Kurzüberblick ... 282
10.2.3 Dateiinhalte verarbeiten und Systemressourcen ... 283
10.3 Praxisbeispiel: Directory-Baum darstellen ... 285
10.3.1 Basisvariante ... 286
10.3.2 Variante mit schönerer Darstellung ... 288
10.3.3 Finale Variante mit ausgeklügelter Darstellung ... 289
10.4 Aufgaben und Lösungen ... 291
10.4.1 Aufgabe 1: Texte in Datei schreiben und wieder lesen ... 291
10.4.2 Aufgabe 2: Dateigrößen ... 291
10.4.3 Aufgabe 3: Existenzprüfung ... 292
10.4.4 Aufgabe 4: Rechteprüfung ... 292
10.4.5 Aufgabe 5: Verzeichnisinhalt auflisten ... 293

11 Fehlerbehandlung mit Exceptions ... 295
11.1 Schnelleinstieg ... 295
11.1.1 Fehlerbehandlung ... 296
11.1.2 Exceptions selbst auslösen – `throw` ... 301
11.1.3 Eigene Exception-Typen definieren ... 302
11.1.4 Exceptions propagieren – `throws` ... 302
11.2 Fehlerbehandlung in der Praxis ... 303
11.3 Automatic Resource Management (ARM) ... 305
11.4 Hintergrundwissen: Checked und Unchecked Exceptions ... 306

12 Datumsverarbeitung ... 309
12.1 Schnelleinstieg ... 309
12.1.1 Die Aufzählungen `DayOfWeek` und `Month` ... 309
12.1.2 Die Klasse `LocalDate` ... 311
12.1.3 Die Klassen `LocalTime` und `LocalDateTime` ... 315

12.2 Nächste Schritte 317
12.2.1 Datumsarithmetik 317
12.2.2 Formatierung und Parsing 319
12.3 Praxisbeispiel: Kalenderausgabe 321
12.4 Aufgaben und Lösungen 324
12.4.1 Aufgabe 1: Wochentage 324
12.4.2 Aufgabe 2: Freitag, der 13. 325
12.4.3 Aufgabe 3: Mehrmals Freitag, der 13. 326
12.4.4 Aufgabe 4: Schaltjahre 327

III Praxisbeispiele 329

13 Praxisbeispiel: Tic Tac Toe 331
13.1 Spielfeld initialisieren und darstellen 331
13.2 Setzen der Steine 332
13.3 Prüfen auf Sieg 333
13.4 Bausteine im Einsatz 335

14 Praxisbeispiel: CSV-Highscore-Liste einlesen 337
14.1 Verarbeitung von Spielständen (Highscores) 337
14.2 Extraktion der Daten 339
14.3 Besonderheiten der Implementierung 340

15 Praxisbeispiel: Worträtsel 341
15.1 Applikationsdesign – Vorüberlegungen zur Strukturierung 342
15.2 Einlesen der verfügbaren Wörter 342
15.3 Hilfsdatenstrukturen 344
15.4 Datenmodell 344
15.4.1 Datenspeicherung und Initialisierung 344
15.4.2 Zufällige Wahl von Richtung, Position, Wort und Buchstabe . . 346
15.4.3 Algorithmus zum Verstecken von Wörtern 346
15.4.4 Wort prüfen und platzieren 348
15.5 HTML-Erzeugung 350
15.6 Hauptapplikation 351
15.7 Ausgabe als HTML und Darstellung im Browser 353
15.8 Fazit 354

IV Schlussgedanken 355

16 Gute Angewohnheiten 357
16.1 Grundregeln eines guten Programmierstils 357
16.2 Coding Conventions 358
16.2.1 Grundlegende Namens- und Formatierungsregeln 358
16.2.2 Namensgebung 360
16.2.3 Dokumentation 363
16.2.4 Programmdesign 363
16.2.5 Parameterlisten 364
16.2.6 Logik und Kontrollfluss 365
16.3 Sourcecode-Prüfung 366
16.4 JUnit 5: Auch ans Testen denken 368
16.4.1 Das JUnit-Framework 368
16.4.2 Schreiben und Ausführen von Tests 370

17 Schlusswort 373

V Anhang 375

A Schlüsselwörter im Überblick 377

B Schnelleinstieg JShell 381

C Grundlagen zur JVM und Infos zum Java-Ökosystem 385
C.1 Wissenswertes zur Java Virtual Machine (JVM) 385
C.1.1 Einführendes Beispiel 386
C.1.2 Ausführung eines Java-Programms 387
C.2 Das Java-Ökosystem im Kurzüberblick 388

Literaturverzeichnis 391

Index 393

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. Dabei fangen wir mit den Grundlagen an und bauen Ihr Wissen immer weiter aus, 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

Dies ist ein Buch für Programmierneulinge. Es wendet sich somit 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, die einfach die wunderbare und vielfältige Welt der Programmierung mit Java kennenlernen möchten.

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 – aber für die Lektüre des Buchs ist es hilfreich, wenn Sie

- einigermaßen fit im Installieren von Programmen sind und
- die Kommandozeile 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 auch ein Buch zum Mitmachen. Ich ermutige Sie ganz besonders, 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, also direkt in der JShell einzugeben, oder später im Editor Ihrer IDE.

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

Tipp: Praxistipp

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. Abgerundet werden viele Kapitel mit diversen Aufgaben und zugehörigen Musterlösungen, sodass das zuvor Gelernte direkt anhand neuer Problemstellungen praktiziert und das Wissen vertieft werden kann.

Für Ihren erfolgreichen Weg zur Java-Programmierung gliedert sich das Buch in die vier Teile Einstieg, Aufstieg, Praxisbeispiele und Schlussgedanken.

Im Teil »Einstieg« werden Grundlagen behandelt. Hier empfiehlt es sich wirklich, die Kapitel in der Reihenfolge des Buchs zu lesen, da mit jedem Kapitel neue Grundlagen 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. Der Teil »Praxisbeispiele« verdeutlicht die bisherigen Lerninhalte anhand von vereinfachten, didaktisch aufbereiteten Beispielen aus der Praxis. Hier haben Sie viel Spielraum zum Experimentieren und Einbringen eigener Ideen. Im Teil »Schlussgedanken« wird ein Ausblick gegeben, etwa auf Programmierstil und Testen. Das Buch endet dann mit einem Rückblick und Hinweisen für nächste Schritte.

Einstieg

Kapitel 1 – Einführung Dieses Kapitel gibt zunächst einen kurzen Überblick über Javas mittlerweile über 25-jährige Geschichte. Bevor wir dann mit dem Lernen von Java als Programmiersprache loslegen können, müssen wir ein paar Installationen vornehmen. Zum einen benötigen wir Java an sich und zum anderen wird eine IDE (Integrated Development Environment) im Verlauf des Buchs mit zunehmender Komplexität der Beispiele immer nützlicher.

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

Kapitel 3 – Strings Strings sind aus kaum einem Programm wegzudenken. Variablen vom Typ `String` repräsentieren Zeichenketten und dienen zur Verwaltung von textuellen Informationen. In diesem Kapitel behandle ich die Thematik genauer.

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 auch 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 Collections oder Containerklassen mehr Flexibilität und Komfort bei der Verwaltung von Daten. In Java unterstützen die vordefinierten Listen, Mengen und Schlüssel-Wert-Abbildungen bei der Verwaltung anderer Objekte.

Kapitel 7 – Ergänzendes Wissen In diesem Kapitel werden verschiedene wichtige Themen angesprochen, die in den vorherigen Kapiteln aus didaktischen Gründen bewusst ausgelassen wurden. Warum? Deren Beschreibung erfordert mehr Wissen rund um Java, was Sie mittlerweile erworben haben – vorher wäre das Ganze aber zu tief in die Details gegangen und hätte zu viel anderes Wissen vorausgesetzt. Hier angelangt lohnt es sich aber, das bisherige Wissen etwa mit Informationen zu primitiven Typen, dem Ternary-Operator, Fallunterscheidungen mit `switch` usw. zu komplettieren.

Aufstieg

Kapitel 8 – Mehr zu Klassen und Objektorientierung Die objektorientierte Programmierung ist ein vielschichtiges und umfangreiches Thema. Kapitel 5 hat eine Einführung geliefert. In diesem Kapitel wird Ihr Wissen noch ein wenig vertieft, beispielsweise zu Besonderheiten in Java, zu Vererbung und Polymorphie.

Kapitel 9 – 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 10 – Verarbeitung von Dateien Dieses Kapitel beschäftigt sich mit der Verarbeitung von Informationen aus Dateien. Dies ist für viele Anwendungen von großer Bedeutung, da diverse Informationen nicht nur während der Programmlaufzeit von Interesse sind, sondern vor allem auch darüber hinaus – denken Sie etwa an die Highscore-Liste Ihres Lieblingsspiels.

Kapitel 11 – Fehlerbehandlung mit Exceptions Sicher kennen Sie es: Manchmal tritt ein Programmfehler auf und das Programm stürzt ab. Wichtige Daten gehen mitunter verloren. So etwas ist immer ärgerlich. Daher gehört auch die Behandlung von Fehlern zum guten Ton beim Programmieren. Diese Kapitel führt in die Thematik ein.

Kapitel 12 – Datumsverarbeitung Während früher die Datumsverarbeitung eher stiefmütterlich in Java unterstützt wurde, bietet modernes Java mittlerweile eine Vielzahl praktischer Funktionalitäten zur Datumsverarbeitung, die in diesem Kapitel einführend dargestellt werden.

Praxisbeispiele

Kapitel 13 – Praxisbeispiel: Tic Tac Toe In diesem Kapitel entwickeln wir eine einfache Version des Strategiespiels Tic Tac Toe, das auf einem Spielfeld mit 3×3 Feldern gespielt wird. Dabei wird verdeutlicht, warum wir Programme sinnvoll in kleine Methodenbausteine untergliedern sollten.

Kapitel 14 – Praxisbeispiel: CSV-Highscore-Liste einlesen In diesem Kapitel verdeutlicht ein weiteres Praxisbeispiel die Verarbeitung von Dateien und kommaseparierter Daten, auch CSV (Comma Separated Values) genannt. Um das Ganze unterhaltsam zu gestalten, werden wir statt trockener Anwendungsdaten eine Liste von Spielständen als Eingabe nutzen.

Kapitel 15 – Praxisbeispiel: Worträtsel Dieses dritte Praxisbeispiel umfasst eine etwas komplexere Programmieraufgabe, nämlich die Erstellung von Worträtseln, die man aus Zeitschriften kennt. Dabei sollen aus einem scheinbaren »Buchstabensalat« verschiedene dort versteckte Begriffe extrahiert werden. Dieses Kapitel vermittelt, wie man Aufgaben in verschiedene kleine Problemstellungen untergliedert und diese jeweils mit eigenen Klassen realisieren kann. Schließlich ist es dann Aufgabe der eigentlichen Applikation, wie ein Dirigent zu wirken und die Einheiten passend zusammenzufügen. Dabei lernen wir beispielsweise den Import möglicher Wörter aus Dateien, die Modellierung des Rätsels und einen einfachen Export nach HTML kennen.

Schlussgedanken

Kapitel 16 – Gute Angewohnheiten Dieses Kapitel stellt Ihnen ein paar Dinge zu gutem Programmierstil vor. Das mündet dann in sogenannten Coding Conventions, also Regeln beim Programmieren. Außerdem zeige ich noch, wie sich einige davon mit Tools prüfen lassen und wie man Programme mit JUnit 5 testen und dadurch Fehler vermeiden kann.

Kapitel 17 – Schlusswort Hier rekapitulieren wir kurz, was Sie durch die Lektüre 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. Der Grund ist vor allem, dass Java seit Version 9 die interaktive Kommandozeilenapplikation JShell als REPL (Read-Eval-Print-Loop) bietet, die in den letzten Java-Versionen immer komfortabler geworden ist.

Anhang C – Grundlagen zur JVM und Infos zum Java-Ökosystem In diesem Anhang vermittle ich vertiefendes Grundwissen zur JVM (Java Virtual Machine). Zudem beleuchte ich kurz das breitgefächerte Ökosystem rund um Java.

Sourcecode und ausführbare Programme

Ich hatte schon angedeutet, dass es zum Erlernen des Programmierens ziemlich hilfreich ist, die Beispiele und Codeschnipsel auch 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 `www.dpunkt.de/Einfach-Java` zur Verfügung. Weitere Informationen zum genauen Vorgehen finden Sie auf der Download-Seite.

Blockkommentare in Listings Beachten Sie bitte, dass sich in einigen Listings mitunter Blockkommentare (hier fett markiert) finden, die der Orientierung und dem besseren Verständnis dienen. In der Praxis sollte man derartige Kommentierungen mit Bedacht einsetzen und lieber einzelne Sourcecode-Abschnitte in Methoden auslagern, wie dies später im Rahmen der Praxisbeispiele offensichtlich wird. Für diverse Beispiele dieses Buchs dienen diese Kommentare aber als Anhaltspunkte, weil die eingeführten oder dargestellten Sachverhalte für Sie als Leser vermutlich noch neu und ungewohnt sind.

```
public static void main(String[] args) throws InterruptedException,
                                               IOException
{
    // Prozess erzeugen
    String command = "sleep 60s";
    Process sleeper = Runtime.getRuntime().exec(command);
    ...

    // Process => ProcessHandle
    ProcessHandle sleeperHandle = ...
    ...
}
```

Konventionen

Verwendete Zeichensätze

In diesem Buch gelten folgende Konventionen bezüglich der Schriftart: 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 dies einen Ausschnitt aus einem Java-Programm darstellt. 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

Verwendete Java-Version(en)

Nahezu alle Beispiele wurden mit Java 16 entwickelt und ausprobiert. 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. Deswegen basieren die meisten Programme auf Java 11, das zudem im Sommer 2021 die aktuelle LTS-Version (Long Term Support) ist. Die neueren Java-Versionen offerieren jedoch einige hilfreiche Syntaxänderungen und API-Erweiterungen. Daher beschreibe und verwende ich diese, wo es sinnvoll ist und das Verständnis erleichtert. In einem Abschnitt werden sogar Neuerungen aus dem im September 2021 erscheinenden Java 17 im Bereich von `switch` vorgestellt.

Somit sind Sie nach der Lektüre dieses Buchs bestens gerüstet für modernes Java und können nach Lust und Laune die neuen Features in eigenen Experimenten und Hobbyprojekten einsetzen.

Danksagung

Wie schon bei einigen meiner bisherigen Bücher hat mich auch diesmal Michael Kulla wieder ganz besonders unterstützt, wie üblich breitgefächert vom Aufdecken von Tippfehlern bis hin zu diversen inhaltlichen Hinweisen. Das gilt ebenfalls für Prof. Dr. Dominik Gruntz. Er hat wie gewohnt mit Spürsinn gelesen und eine Vielzahl an hilfreichen Anmerkungen hinterlassen. Auch Jean-Claude Brantschen trug mit seinen Kommentaren und Tipps zur Verbesserung bei. Außerdem gilt Christian Heitzmann ein herzlicher Dank für diverse Anregungen zum Inhalt, zur Struktur sowie zu Begrifflichkeiten. Schließlich hat Maria Herdt freundlicherweise durch einen kritischen Blick auf einige Kapitel ein paar Verbesserungen aufzeigen können. Nochmals vielen Dank an euch alle!

Zunächst geht ein Dankeschön an das Team des dpunkt.verlags (Dr. Michael Barabas, Anja Weimer, Stefanie Weidner und Veronika Schnabel) 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 Juli 2021
Michael Inden

Einstieg

1 Einführung

1.1 Java im Überblick

Java wurde Mitte der 1990er-Jahre von der Firma Sun entwickelt und später von Oracle übernommen. Mittlerweile hat Java zwar schon mehr als 25 Jahre auf dem Buckel, wird aber nicht altersschwach, sondern kontinuierlich gepflegt und besitzt ein extrem breitgefächertes und professionelles Angebot an externen Bibliotheken und Entwicklungstools. Insbesondere gibt es mit Eclipse, IntelliJ IDEA und NetBeans sowie Visual Studio Code mehrere hervorragende sogenannte IDEs (Integrated Development Environments) zum komfortablen Programmieren.

Java eignet sich zur Entwicklung unterschiedlichster Applikationen, etwa für Businessapplikationen, Webapplikationen und sogar einige (einfachere) Datenbanken. Es wird aber auch im Bereich Mobile in Form von Android-Apps oder gar im Bereich von Spielen, z. B. Minecraft, verwendet. Java ist also eine vielseitige Programmiersprache mit breitem Einsatzspektrum. Ein guter Grund, sich damit ein wenig zu beschäftigen.

Außerdem ist Programmieren ein wunderbares Hobby sowie ein faszinierender Beruf und es macht zudem noch jede Menge Spaß, fördert die Kreativität und den Gestaltungswillen.

Darüber hinaus ist Java laut TIOBE-Index[1] seit Jahren eine der populärsten Programmiersprachen. Eine wichtige Rolle spielte vermutlich lange die freie Verfügbarkeit. Zumindest für die Originalvariante von Oracle gilt das mittlerweile nur noch für private, nicht kommerzielle Zwecke. Praktischerweise existieren inzwischen einige frei verwendbare Alternativen wie etwa das AdoptOpenJDK[2]. Zudem ist Java recht einfach zu erlernen (schwieriger als Python, aber deutlich leichter als C++) und bietet ein großes Ökosystem an Tools, Bibliotheken und Literatur sowie Hilfestellungen wie StackOverflow im Internet. Weiterhin zeichnet sich Java durch seine gute Performance aus. Das Ganze ist die Folge von jahrelangen Optimierungen und Verbesserungen. Dadurch ist die Ausführung von Java beispielsweise nur geringfügig langsamer als die von C++, aber um Längen schneller als die Ausführung von Python. Schließlich ermöglicht Java sowohl die objektorientierte als auch die funktionale Programmierung, sodass man je nach Einsatzzweck geeignet wählen kann.

[1] `https://www.tiobe.com/tiobe-index/`

[2] `https://adoptopenjdk.net/`. Ab März 2021 wird das Ganze als Eclipse Adoptium weitergeführt: `https://projects.eclipse.org/projects/adoptium`).

Wie Sie sehen, sprechen viele gute Gründe für einen Einstieg in die Programmierung mit Java. Das Wichtigste ist jedoch der Spaß am Programmieren, Tüfteln und Ausprobieren. Lassen Sie uns starten!

Bestandteile von Java-Programmen

Java als Programmiersprache besitzt wie eine natürliche Sprache auch eine Grammatik und feststehende Begriffe / Wörter. Man spricht dabei von Syntax und Schlüsselwörtern (vgl. Anhang A).

Java-Programme werden textuell verfasst. Das wird ***Sourcecode*** genannt. Schauen wir uns zum Einstieg ein einfaches Java-Programm an:

```
public class MyFirstJavaProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

Keine Sorge, Sie müssen das Ganze noch nicht vollständig verstehen, wir werden das alles Stück für Stück erlernen. Hier ist zunächst nur wichtig, dass Sie elementare Bestandteile von Java-Programmen grob einordnen können. Dazu gehören die Schlüsselwörter, also Java-Befehle oder -Anweisungen, hier etwa `public`, `class`, `static` und `void`. Wie die Begriffe in einer Sprache tragen diese reservierten Wörter eine besondere Bedeutung, ganz analog etwa zu Auto, Haus, Tür usw. im Deutschen.

Ebenso wie im Deutschen können (oder besser sollten) die Begriffe nicht einfach wahllos miteinander verknüpft werden, um einen gültigen Satz zu formulieren. Das wird durch die Grammatik geregelt. Auch in Java existiert eine solche. Damit wird etwa festgelegt, dass es `static void`, aber nicht `void static` heißen muss.

Zudem sehen wir geschweifte Klammern. Diese kann man sich wie Absätze in einem Text vorstellen. In Java bündeln diese Klammern Anweisungen. Man spricht dann auch von Blöcken und Sichtbarkeitsbereichen.

Genug der Vielzahl an Informationen. Nachfolgend werden wir die Dinge schön gründlich und detailliert besprechen und didaktisch immer ein neues Themengebiet ergründen, bis wir schließlich einen guten Einstieg in die Java-Programmierung vollzogen haben werden.

Vorab wollen wir aber erst einmal Java und Eclipse installieren und erste Schritte machen, um für unsere weitere Entdeckungsreise bereit zu sein.

1.2 Los geht's – Installation

Im ersten Teil dieses Buchs wird ein Hands-on-Ansatz verfolgt, bei dem wir Dinge oftmals in Form kleinerer Java-Codeschnipsel direkt ausprobieren. Sie benötigen vorab keine tiefgreifenden Programmiererfahrungen, allerdings schaden diese natürlich nicht, ganz im Gegenteil. Hilfreich wäre allerdings, wenn Sie sich einigermaßen mit dem Installieren von Programmen und grundlegend mit der Kommandozeile auskennen.

Damit Sie die nachfolgend beschriebenen Java-Programme ausführen können, benötigen Sie ein sogenanntes JDK (Java Development Kit). Dort finden sich alle für den Moment benötigten Tools. Beginnen wir also mit der Installation von Java.

1.2.1 Java-Download

Die aktuelle Java-Version ist frei auf der folgenden Oracle-Webseite verfügbar: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

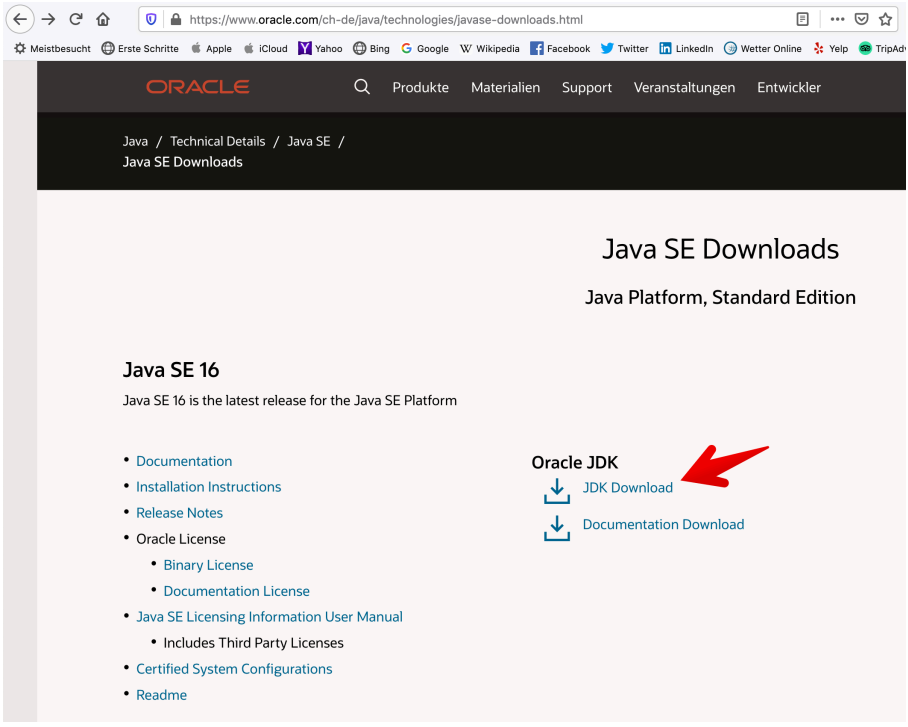

Abbildung 1-1 *Oracle-Seite für Java*

Nachdem Sie auf den Link zum JDK-Download geklickt haben, erscheint in etwa eine Darstellung wie die folgende:

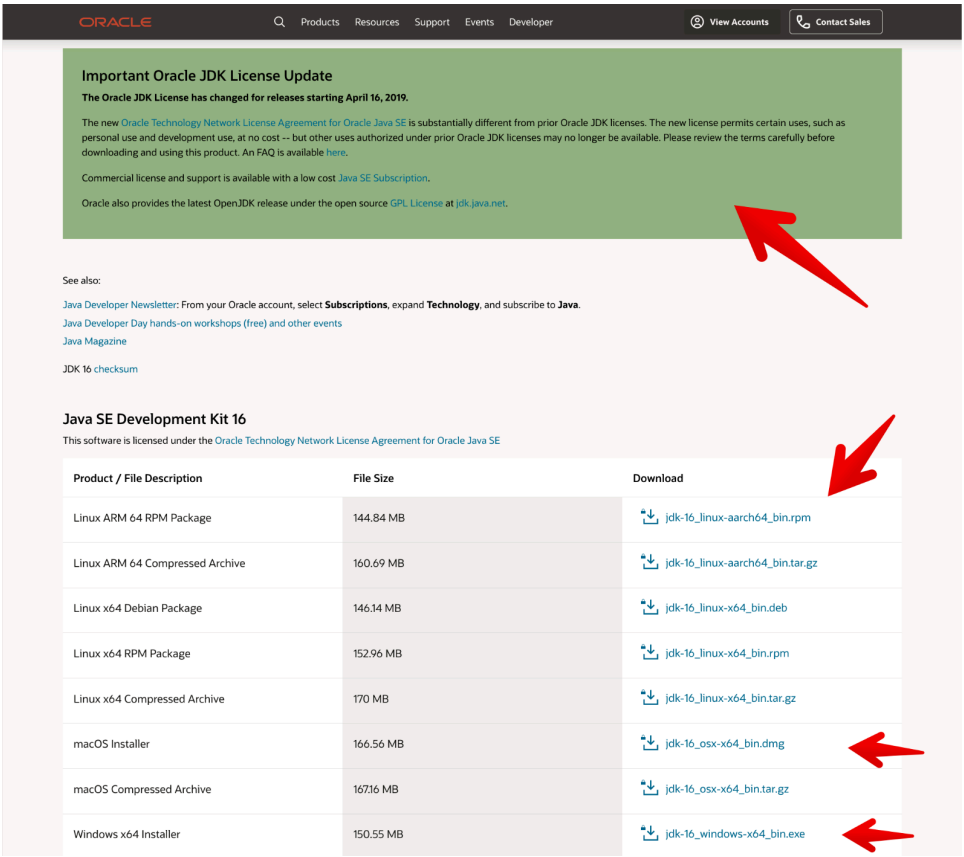

Abbildung 1-2 *Java-Download-Seite*

Im oberen Bereich sehen wir einige Hinweise zur neuen Lizenzpolitik, die ich nachfolgend im Hinweiskasten thematisiere. Im unteren Bereich finden Sie verschiedene Links für unterschiedliche Betriebssysteme. Wählen Sie den für Sie passenden Link.

Hinweis: Neue Lizenzpolitik bei Oracle

Wenn Sie Ihre Software kommerziell vertreiben oder dies planen, dann sollten Sie beim Herunterladen von Java unbedingt die Lizenzpolitik von Oracle beachten: **Das bis Java 8 selbst in Produktionssystemen immer kostenfrei verwendbare Oracle JDK ist dafür nun leider kostenpflichtig**. Als Alternative können Sie auf das OpenJDK (`https://openjdk.java.net/`) ausweichen. Für dieses Buch das Wichtigste: ***Für private Projekte und während der Entwicklung kann das Oracle JDK weiterhin kostenfrei genutzt werden***.

1.2.2 Installation des JDKs

Unter MacOS doppelklicken Sie auf die `.dmg`-Datei, um die Installationsdatei zu starten, und folgen den Aufforderungen. Möglicherweise müssen Sie das Administrator-Passwort eingeben, um fortzufahren. Nachdem die Installation abgeschlossen ist, können Sie die `.dmg`-Datei löschen, um Speicherplatz zu sparen.

Für Windows doppelklicken Sie bitte auf die `.exe`-Datei. Auch diese kann nach erfolgreicher Installation gelöscht werden. Führen Sie also das heruntergeladene Installationsprogramm aus (z. B. `jdk-16.0.1_windows-x64_bin.exe`). Damit wird Java installiert – standardmäßig ins Verzeichnis `C:\Programme\Java\jdk-16.0.1`, wobei der Verzeichnisname von der gewählten Version abhängt. Akzeptieren Sie die Standardeinstellungen und befolgen Sie die Anweisungen während der Installation.

1.2.3 Nacharbeiten nach der Java-Installation

Damit Java bei Ihnen nach dem Download und der Installation auch in der Konsole korrekt funktioniert, sind noch ein paar Nacharbeiten nötig. Dazu müssen wir es zur leichteren Handhabung in den Pfad aufnehmen. Dies wird im Anschluss für die weitverbreiteten Betriebssysteme Windows und MacOS beschrieben. Falls Sie ein Unix-Derivat nutzen, dann finden Sie weitere Informationen auf dieser Seite: `https://www.java.com/de/download/help/download_options.html`. Für Windows und Mac gibt es dort auch noch einige ergänzende Informationen.

Nacharbeiten für Windows

Das Installationsverzeichnis muss in die Umgebungsvariable `PATH` aufgenommen werden. Diese können Sie unter »Umgebungsvariablen« ändern. Drücken Sie die Win-Taste und geben Sie dann »umgeb« ein, bis »Systemumgebungsvariablen bearbeiten« erscheint. Mit Enter erscheint der Dialog »Systemvariablen«. Klicken Sie auf den Button »Bearbeiten« zum Öffnen eines Bearbeitungsdialogs. Fügen Sie in der Liste das Installationsverzeichnis gefolgt von `bin`, etwa `C:\Programme\Java\jdk-16.0.1\bin`, hinzu. Um nicht den gesamten Pfad eingeben zu müssen, bietet es sich an, eine weitere Umgebungsvariable namens `JAVA_HOME` anzulegen.

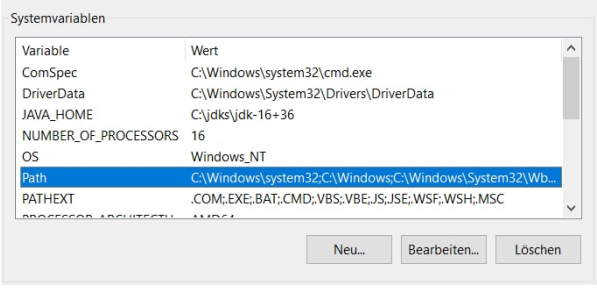

Abbildung 1-3 *Umgebungsvariablen bearbeiten*

Außerdem sollte der Eintrag möglichst ganz oben stehen:

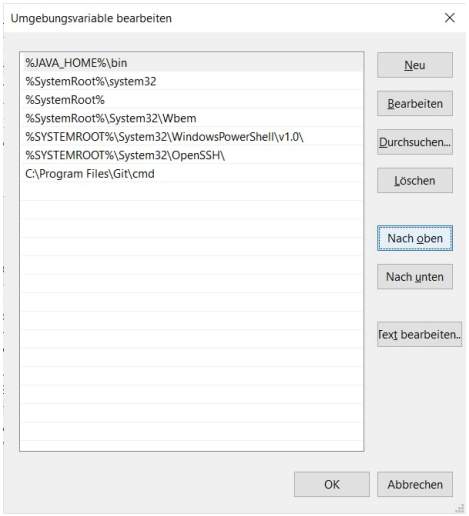

Abbildung 1-4 *Umgebungsvariablen ordnen*

Beachten Sie bitte noch Folgendes: Bestätigen Sie die gesamten Dialoge bitte immer mit OK, sodass die Variablen gesetzt sind. Eventuell geöffnete Konsolen müssen geschlossen und neu geöffnet werden, um die geänderten Variablen wirksam werden zu lassen.

Nacharbeiten für MacOS

Auch unter MacOS empfiehlt es sich, einen Verweis auf Java im Pfad in der jeweiligen Shell (dem Terminal) passend zu setzen.

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-16.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:$PATH
```

1.2.4 Java-Installation prüfen

Nach dem Ausführen der obigen Schritte sollte Java auf Ihrem Rechner installiert und von der Konsole startbar sein und Sie damit bereit für die nächsten Schritte.

Öffnen Sie eine Konsole und geben Sie folgendes Kommando ein – im folgenden Text nutze ich immer $ zur Kennzeichnung von Eingaben auf der Konsole, also dem Terminal bei MacOS bzw. der Windows-Eingabeaufforderung:

```
$ java --version
java 16 2021-03-16
Java(TM) SE Runtime Environment (build 16+36-2231)
Java HotSpot(TM) 64-Bit Server VM (build 16+36-2231, mixed mode, sharing)
```