

5.

Auflage



Michael Inden

# Der Weg zum Java-Profi

Konzepte und Techniken für die  
professionelle Java-Entwicklung

**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**

# **Der Weg zum Java-Profi**

**Konzepte und Techniken für die  
professionelle Java-Entwicklung**

5., überarbeitete und aktualisierte Auflage



**dpunkt.verlag**

Michael Inden  
*michael\_inden@hotmail.com*

Lektorat: Dr. Michael Barabas  
Projektkoordinierung/Lektoratsassistentz: Anja Weimer  
Fachgutachten: Torsten Horn, Aachen  
Copy-Editing: Ursula Zimpfer, Herrenberg  
Satz: Michael Inden  
Herstellung: Stefanie Weidner  
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*  
Druck: Schleunungdruck GmbH, Marktheidenfeld  
Bindung: Hubert & Co. GmbH & Co. KG. BuchPartner, Göttingen

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 978-3-86490-707-4  
PDF 978-3-96088-842-0  
ePub 978-3-96088-843-7  
mobi 978-3-96088-844-4

5., überarbeitete und aktualisierte Auflage 2021  
Copyright © 2021 dpunkt.verlag GmbH  
Wieblinger Weg 17  
69123 Heidelberg

*Hinweis:*  
Der Umwelt zuliebe verzichten wir auf die Einschweißfolie.

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

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

---

# Inhaltsübersicht

1	Einleitung .....	1
<b>I</b>	<b>Java-Grundlagen, Analyse und Design</b>	<b>11</b>
2	Professionelle Arbeitsumgebung .....	13
3	Objektorientiertes Design .....	85
4	Lambdas, Methodenreferenzen und Defaultmethoden .....	199
5	Java-Grundlagen .....	221
<b>II</b>	<b>Bausteine stabiler Java-Applikationen</b>	<b>325</b>
6	Das Collections-Framework .....	327
7	Das Stream-API .....	453
8	Datumsverarbeitung seit JDK 8 .....	497
9	Applikationsbausteine .....	529
10	Multithreading-Grundlagen .....	581
11	Modern Concurrency .....	633
12	Fortgeschrittene Java-Themen .....	681
13	Basiswissen Internationalisierung .....	743

<b>III</b>	<b>Wichtige Neuerungen in Java 12 bis 15</b>	<b>771</b>
14	Neues und Änderungen in den Java-Versionen 12 bis 15 .....	773
<b>IV</b>	<b>Modularisierung</b>	<b>823</b>
15	Modularisierung mit Project Jigsaw .....	825
<b>V</b>	<b>Fallstricke und Lösungen im Praxisalltag</b>	<b>877</b>
16	Bad Smells .....	879
17	Refactorings .....	953
18	Entwurfsmuster .....	1023
<b>VI</b>	<b>Qualitätssicherungsmaßnahmen</b>	<b>1099</b>
19	Programmierstil und Coding Conventions .....	1101
20	Unit Tests .....	1143
21	Codereviews .....	1265
22	Optimierungen .....	1273
23	Schlussgedanken .....	1349
<b>VII</b>	<b>Anhang</b>	<b>1351</b>
A	Grundlagen zur Java Virtual Machine .....	1353
	Literaturverzeichnis .....	1357
	Index .....	1361

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b> .....	<b>1</b>
1.1	Über dieses Buch .....	1
1.1.1	Motivation .....	1
1.1.2	Was leistet dieses Buch und was nicht? .....	2
1.1.3	Wie und was soll mithilfe des Buchs gelernt werden? .....	2
1.1.4	Wer sollte dieses Buch lesen? .....	4
1.2	Aufbau des Buchs .....	4
1.3	Konventionen und ausführbare Programme .....	6
<b>I Java-Grundlagen, Analyse und Design</b>		<b>11</b>
<b>2</b>	<b>Professionelle Arbeitsumgebung</b> .....	<b>13</b>
2.1	Vorteile von IDEs am Beispiel von Eclipse .....	14
2.2	Projektorganisation .....	16
2.2.1	Projektstruktur in Eclipse .....	16
2.2.2	Projektstruktur für Maven und Gradle .....	18
2.3	Einsatz von Versionsverwaltungen .....	20
2.3.1	Arbeiten mit zentralen Versionsverwaltungen .....	23
2.3.2	Dezentrale Versionsverwaltungen .....	28
2.3.3	VCS und DVCS im Vergleich .....	34
2.4	Einsatz eines Unit-Test-Frameworks .....	37
2.4.1	Das JUnit-Framework .....	37
2.4.2	Parametrierte Tests mit JUnit 5 .....	43
2.4.3	Vorteile von Unit Tests .....	45
2.5	Debugging .....	46
2.5.1	Fehlersuche mit einem Debugger .....	47
2.5.2	Remote Debugging .....	50
2.6	Deployment von Java-Applikationen .....	55
2.6.1	Das JAR-Tool im Kurzüberblick .....	56
2.6.2	JAR inspizieren und ändern, Inhalt extrahieren .....	57
2.6.3	Metainformationen und das Manifest .....	58
2.6.4	Inspizieren einer JAR-Datei .....	60

2.7	Einsatz eines IDE-unabhängigen Build-Prozesses .....	63
2.7.1	Maven im Überblick .....	65
2.7.2	Builds mit Gradle .....	70
2.7.3	Vorteile von Maven und Gradle .....	82
2.8	Weiterführende Literatur .....	83
<b>3</b>	<b>Objektorientiertes Design .....</b>	<b>85</b>
3.1	OO-Grundlagen .....	86
3.1.1	Grundbegriffe .....	86
3.1.2	Beispielentwurf: Ein Zähler .....	98
3.1.3	Vom imperativen zum objektorientierten Entwurf .....	106
3.1.4	Diskussion der OO-Grundgedanken .....	111
3.1.5	Wissenswertes zum Objektzustand .....	114
3.2	Grundlegende OO-Techniken .....	123
3.2.1	Schnittstellen (Interfaces) .....	123
3.2.2	Basisklassen und abstrakte Basisklassen .....	128
3.2.3	Interfaces und abstrakte Basisklassen .....	130
3.3	Wissenswertes zu Vererbung .....	132
3.3.1	Probleme durch Vererbung .....	132
3.3.2	Delegation statt Vererbung .....	138
3.4	Fortgeschrittenere OO-Techniken .....	142
3.4.1	Read-only-Interface .....	142
3.4.2	Immutable-Klasse .....	148
3.4.3	Marker-Interface .....	153
3.4.4	Konstantensammlungen und Aufzählungen .....	154
3.4.5	Value Object (Data Transfer Object) .....	160
3.5	Prinzipien guten OO-Designs .....	162
3.5.1	Geheimnisprinzip nach Parnas .....	163
3.5.2	Law of Demeter .....	164
3.5.3	SOLID-Prinzipien .....	167
3.6	Formen der Varianz .....	178
3.6.1	Grundlagen der Varianz .....	178
3.6.2	Kovariante Rückgabewerte .....	182
3.7	Generische Typen (Generics) .....	184
3.7.1	Einführung .....	184
3.7.2	Generics und Auswirkungen der Type Erasure .....	189
3.8	Weiterführende Literatur .....	197
<b>4</b>	<b>Lambdas, Methodenreferenzen und Defaultmethoden .....</b>	<b>199</b>
4.1	Einstieg in Lambdas .....	199
4.1.1	Syntax von Lambdas .....	199
4.1.2	Functional Interfaces und SAM-Typen .....	200
4.1.3	Exceptions in Lambdas .....	204

4.2	Syntaxerweiterungen in Interfaces .....	208
4.2.1	Defaultmethoden .....	208
4.2.2	Statische Methoden in Interfaces .....	211
4.3	Methodenreferenzen .....	213
4.4	Externe vs. interne Iteration .....	214
4.5	Wichtige Functional Interfaces für Collections .....	215
4.5.1	Das Interface <code>Predicate&lt;T&gt;</code> .....	215
4.5.2	Das Interface <code>UnaryOperator&lt;T&gt;</code> .....	218
4.6	Praxiswissen: Definition von Lambdas .....	219
<b>5</b>	<b>Java-Grundlagen .....</b>	<b>221</b>
5.1	Die Klasse <code>Object</code> .....	221
5.1.1	Die Methode <code>toString()</code> .....	223
5.1.2	Die Methode <code>equals()</code> .....	227
5.2	Primitive Typen und Wrapper-Klassen .....	239
5.2.1	Grundlagen .....	240
5.2.2	Konvertierung von Werten .....	247
5.2.3	Wissenswertes zu Auto-Boxing und Auto-Unboxing .....	250
5.2.4	Ausgabe und Verarbeitung von Zahlen .....	253
5.3	Stringverarbeitung .....	256
5.3.1	Die Klasse <code>String</code> .....	257
5.3.2	Die Klassen <code>StringBuffer</code> und <code>StringBuilder</code> .....	261
5.3.3	Ausgaben mit <code>format()</code> und <code>printf()</code> .....	264
5.3.4	Die Methode <code>split()</code> und reguläre Ausdrücke .....	265
5.3.5	Optimierung bei Strings in JDK 9 .....	270
5.3.6	Neue Methoden in der Klasse <code>String</code> in JDK 11 .....	270
5.4	Varianten innerer Klassen .....	272
5.5	Ein- und Ausgabe (I/O) .....	276
5.5.1	Dateibehandlung und die Klasse <code>File</code> .....	276
5.5.2	Ein- und Ausgabestreams im Überblick .....	280
5.5.3	Zeichencodierungen bei der Ein- und Ausgabe .....	283
5.5.4	Speichern und Laden von Daten und Objekten .....	285
5.5.5	Dateiverarbeitung mit dem NIO .....	293
5.5.6	Neue Hilfsmethoden in der Klasse <code>Files</code> in JDK 11 .....	296
5.6	Fehlerbehandlung .....	298
5.6.1	Einstieg in die Fehlerbehandlung .....	298
5.6.2	Checked Exceptions und Unchecked Exceptions .....	304
5.6.3	Besonderheiten beim Exception Handling .....	306
5.6.4	Exception Handling und Ressourcenfreigabe .....	310
5.6.5	Assertions .....	314

5.7	Weitere Neuerungen in JDK 9, 10 und 11 .....	318
5.7.1	Erweiterung der <code>@Deprecated</code> -Annotation in JDK 9 .....	318
5.7.2	Syntaxerweiterung <code>var</code> in JDK 10 und 11 .....	319
5.7.3	Versionsverarbeitung mit JDK 9 und 10 .....	323
5.8	Weiterführende Literatur .....	324

## **II Bausteine stabiler Java-Applikationen** **325**

<b>6</b>	<b>Das Collections-Framework .....</b>	<b>327</b>
6.1	Datenstrukturen und Containerklassen .....	327
6.1.1	Wahl einer geeigneten Datenstruktur .....	328
6.1.2	Arrays .....	330
6.1.3	Das Interface <code>Collection</code> .....	332
6.1.4	Das Interface <code>Iterator</code> .....	334
6.1.5	Listen und das Interface <code>List</code> .....	338
6.1.6	Mengen und das Interface <code>Set</code> .....	346
6.1.7	Grundlagen von hashbasierten Containern .....	350
6.1.8	Grundlagen automatisch sortierender Container .....	359
6.1.9	Die Methoden <code>equals()</code> , <code>hashCode()</code> und <code>compareTo()</code> im Zusammenspiel .....	367
6.1.10	Schlüssel-Wert-Abbildungen und das Interface <code>Map</code> .....	369
6.1.11	Erweiterungen am Beispiel der Klasse <code>HashMap</code> .....	378
6.1.12	Erweiterungen im Interface <code>Map</code> in JDK 8 .....	381
6.1.13	Collection-Factory-Methoden in JDK 9 .....	386
6.1.14	Unveränderliche Kopien von Collections mit Java 10 .....	390
6.1.15	Entscheidungshilfe zur Wahl von Datenstrukturen .....	392
6.2	Suchen und Sortieren .....	393
6.2.1	Suchen .....	393
6.2.2	Sortieren von Arrays und Listen .....	396
6.2.3	Sortieren mit Komparatoren .....	399
6.2.4	Erweiterungen im Interface <code>Comparator</code> mit JDK 8 .....	401
6.3	Utility-Klassen und Hilfsmethoden .....	406
6.3.1	Nützliche Hilfsmethoden .....	406
6.3.2	Dekorierer <code>synchronized</code> und <code>unmodifiable</code> .....	411
6.3.3	Vordefinierte Algorithmen in der Klasse <code>Collections</code> ...	415
6.4	Containerklassen: Generics und Varianz .....	417
6.5	Die Klasse <code>Optional&lt;T&gt;</code> .....	431
6.5.1	Grundlagen zur Klasse <code>Optional</code> .....	432
6.5.2	Weiterführendes Beispiel und Diskussion .....	436
6.5.3	Verkettete Methodenaufrufe .....	438
6.5.4	Erweiterungen in der Klasse <code>Optional&lt;T&gt;</code> in JDK 9 .....	439
6.5.5	Erweiterung in <code>Optional&lt;T&gt;</code> in JDK 10 und 11 .....	442

6.6	Fallstricke im Collections-Framework .....	443
6.6.1	Wissenswertes zu Arrays .....	443
6.6.2	Wissenswertes zu Stack, Queue und Deque .....	447
6.7	Weiterführende Literatur .....	450
<b>7</b>	<b>Das Stream-API .....</b>	<b>453</b>
7.1	Grundlagen zu Streams .....	453
7.1.1	Streams erzeugen – Create Operations .....	454
7.1.2	Intermediate und Terminal Operations im Überblick .....	456
7.1.3	Zustandslose Intermediate Operations .....	458
7.1.4	Zustandsbehaftete Intermediate Operations .....	465
7.1.5	Terminal Operations .....	467
7.1.6	Wissenswertes zur Parallelverarbeitung .....	476
7.1.7	Neuerungen im Stream-API in JDK 9 .....	481
7.1.8	Neuerungen im Stream-API in JDK 10 .....	484
7.2	Filter-Map-Reduce .....	485
7.2.1	Herkömmliche Realisierung .....	486
7.2.2	Filter-Map-Reduce mit JDK 8 .....	488
7.3	Praxisbeispiele .....	490
7.3.1	Aufbereiten von Gruppierungen und Histogrammen .....	491
7.3.2	Maps nach Wert sortieren .....	492
<b>8</b>	<b>Datumsverarbeitung seit JDK 8 .....</b>	<b>497</b>
8.1	Überblick über die neu eingeführten Typen .....	497
8.1.1	Neue Aufzählungen, Klassen und Interfaces .....	498
8.1.2	Die Aufzählungen DayOfWeek und Month .....	500
8.1.3	Die Klassen MonthDay, YearMonth und Year .....	500
8.1.4	Die Klasse Instant .....	501
8.1.5	Die Klasse Duration .....	502
8.1.6	Die Aufzählung ChronoUnit .....	506
8.1.7	Die Klassen LocalDate, LocalTime und LocalDateTime .....	507
8.1.8	Die Klasse Period .....	510
8.1.9	Die Klasse ZonedDateTime .....	511
8.1.10	Zeitzonen und die Klassen ZoneId und ZoneOffset .....	512
8.1.11	Die Klasse Clock .....	514
8.1.12	Formatierung und Parsing .....	516
8.2	Datumsarithmetik .....	517
8.2.1	Einstieg in die Datumsarithmetik .....	517
8.2.2	Real-World-Example: Gehaltszahltag .....	522
8.3	Interoperabilität mit Legacy-Code .....	526

<b>9</b>	<b>Applikationsbausteine</b> .....	<b>529</b>
9.1	Einsatz von Bibliotheken .....	530
9.2	Google Guava im Kurzüberblick .....	533
9.2.1	String-Aktionen .....	534
9.2.2	Stringkonkatenation und -extraktion .....	535
9.2.3	Erweiterungen für Collections .....	540
9.2.4	Weitere Utility-Funktionalitäten .....	544
9.3	Wertebereichs- und Parameterprüfungen .....	548
9.4	Logging-Frameworks .....	551
9.4.1	Apache log4j2 .....	552
9.4.2	Tipps und Tricks zum Einsatz von Logging mit log4j2 .....	556
9.5	Konfigurationsparameter und -dateien .....	561
9.5.1	Einlesen von Kommandozeilenparametern .....	561
9.5.2	Verarbeitung von Properties .....	569
9.5.3	Weitere Möglichkeiten zur Konfigurationsverwaltung .....	575
<b>10</b>	<b>Multithreading-Grundlagen</b> .....	<b>581</b>
10.1	Threads und Runnables .....	583
10.1.1	Definition der auszuführenden Aufgabe .....	583
10.1.2	Start, Ausführung und Ende von Threads .....	584
10.1.3	Lebenszyklus von Threads und Thread-Zustände .....	588
10.1.4	Unterbrechungswünsche durch Aufruf von <code>interrupt()</code> .....	591
10.2	Zusammenarbeit von Threads .....	594
10.2.1	Konkurrierende Datenzugriffe .....	594
10.2.2	Locks, Monitore und kritische Bereiche .....	595
10.2.3	Deadlocks und Starvation .....	602
10.2.4	Kritische Bereiche und das Interface <code>Lock</code> .....	604
10.3	Kommunikation von Threads .....	606
10.3.1	Kommunikation mit Synchronisation .....	606
10.3.2	Kommunikation über die Methoden <code>wait()</code> , <code>notify()</code> und <code>notifyAll()</code> .....	610
10.3.3	Abstimmung von Threads .....	615
10.3.4	Unerwartete <code>IllegalMonitorStateException</code> .....	617
10.4	Das Java-Memory-Modell .....	619
10.4.1	Sichtbarkeit .....	619
10.4.2	Atomarität .....	620
10.4.3	Reorderings .....	622
10.5	Besonderheiten bei Threads .....	626
10.5.1	Verschiedene Arten von Threads .....	626
10.5.2	Exceptions in Threads .....	627
10.5.3	Sicheres Beenden von Threads .....	628
10.6	Weiterführende Literatur .....	632

<b>11</b>	<b>Modern Concurrency</b> .....	<b>633</b>
11.1	Concurrent Collections .....	634
11.1.1	Thread-Sicherheit und Parallelität mit »normalen« Collections .....	634
11.1.2	Parallelität mit den Concurrent Collections .....	636
11.1.3	Blockierende Warteschlangen und das Interface <code>BlockingQueue&lt;E&gt;</code> .....	639
11.2	Das Executor-Framework .....	642
11.2.1	Einführung .....	642
11.2.2	Definition von Aufgaben .....	645
11.2.3	Parallele Abarbeitung im <code>ExecutorService</code> .....	649
11.3	Das Fork-Join-Framework .....	652
11.3.1	Einführendes Beispiel .....	653
11.3.2	Real-World-Example: Merge Sort .....	656
11.4	Die Klasse <code>CompletableFuture&lt;T&gt;</code> .....	659
11.4.1	Einführung .....	659
11.4.2	Beispiel: Parallele Verarbeitung von Dateiinhalten .....	662
11.4.3	Erweiterungen in JDK 9 .....	665
11.4.4	Beispiel: Von synchron zu <code>mutlithreaded</code> .....	667
11.5	Reactive Streams und die Klasse <code>Flow</code> .....	670
11.5.1	Schnelleinstieg Reactive Streams .....	670
11.5.2	Reactive Streams im JDK .....	671
11.5.3	Beispiel zur Klasse <code>Flow</code> .....	675
11.5.4	Fazit .....	679
11.6	Weiterführende Literatur .....	680
<b>12</b>	<b>Fortgeschrittene Java-Themen</b> .....	<b>681</b>
12.1	Crashkurs Reflection .....	681
12.1.1	Grundlagen .....	683
12.1.2	Zugriff auf Methoden und Attribute .....	686
12.1.3	Spezialfälle .....	691
12.1.4	Type Erasure und Typinformationen bei Generics .....	697
12.1.5	Fazit .....	699
12.2	Annotations .....	699
12.2.1	Einführung in Annotations .....	700
12.2.2	Standard-Annotations des JDKs .....	701
12.2.3	Definition eigener Annotations .....	703
12.2.4	Annotations zur Laufzeit auslesen .....	706
12.3	Serialisierung .....	708
12.3.1	Grundlagen der Serialisierung .....	708
12.3.2	Die Serialisierung anpassen .....	713
12.3.3	Versionsverwaltung der Serialisierung .....	716
12.3.4	Optimierung der Serialisierung .....	720

12.4	Garbage Collection .....	725
12.4.1	Grundlagen zur Garbage Collection .....	725
12.4.2	Der Garbage Collector »G1« .....	727
12.5	Dynamic Proxies .....	728
12.5.1	Statischer Proxy .....	730
12.5.2	Dynamischer Proxy .....	732
12.6	HTTP/2-API .....	737
12.6.1	Einführung .....	737
12.6.2	Real-World-Example: Wechselkurs mit REST .....	740
12.6.3	Fazit .....	742
12.7	Weiterführende Literatur .....	742
<b>13</b>	<b>Basiswissen Internationalisierung .....</b>	<b>743</b>
13.1	Internationalisierung im Überblick .....	743
13.1.1	Grundlagen und Normen .....	744
13.1.2	Die Klasse <code>Locale</code> .....	745
13.1.3	Die Klasse <code>PropertyResourceBundle</code> .....	749
13.1.4	Formatierte Ein- und Ausgabe .....	752
13.1.5	Datumswerte und die Klasse <code>DateFormat</code> .....	753
13.1.6	Zahlen und die Klasse <code>NumberFormat</code> .....	754
13.1.7	Textmeldungen und die Klasse <code>MessageFormat</code> .....	757
13.1.8	Stringvergleiche mit der Klasse <code>Collator</code> .....	759
13.2	Programmbausteine zur Internationalisierung .....	764
13.2.1	Unterstützung mehrerer Datumsformate .....	765
13.2.2	Fazit und Ausblick .....	770

<b>III</b>	<b>Wichtige Neuerungen in Java 12 bis 15</b>	<b>771</b>
------------	--	------------

<b>14</b>	<b>Neues und Änderungen in den Java-Versionen 12 bis 15 .....</b>	<b>773</b>
14.1	Syntaxneuerungen .....	774
14.1.1	Text Blocks .....	774
14.1.2	Switch Expressions .....	779
14.1.3	Records (Preview) .....	786
14.1.4	Pattern Matching bei <code>instanceof</code> (Preview) .....	793
14.1.5	Sealed Types (Preview) .....	795
14.1.6	Lokale Enums und Interfaces (Preview) .....	797
14.2	API-Neuerungen .....	798
14.2.1	Neue Methoden in der Klasse <code>String</code> .....	798
14.2.2	Neue Hilfsmethode in der Utility-Klasse <code>Files</code> .....	801
14.2.3	Der <code>teeing()</code> -Kollektor .....	802

14.3	JVM-Neuerungen .....	805
14.3.1	Verbesserung bei <code>NullPointerException</code> .....	805
14.3.2	Entfernung der JavaScript-Engine .....	808
14.4	Microbenchmark Suite .....	808
14.4.1	Eigene Microbenchmarks und Varianten davon .....	808
14.4.2	Microbenchmarks mit JMH .....	811
14.4.3	Fazit zu JMH .....	817
14.5	Java 15 – notwendige Anpassungen für Build-Tools und IDEs .....	818
14.5.1	Java 15 mit Gradle .....	818
14.5.2	Java 15 mit Maven .....	819
14.5.3	Java 15 mit Eclipse .....	820
14.5.4	Java 15 mit IntelliJ .....	821
14.5.5	Java 15 mit JShell oder der Kommandozeile .....	821
14.6	Fazit .....	822

<b>IV Modularisierung</b>	<b>823</b>
---------------------------	------------

<b>15</b>	<b>Modularisierung mit Project Jigsaw .....</b>	<b>825</b>
15.1	Grundlagen .....	826
15.1.1	Begrifflichkeiten .....	826
15.1.2	Ziele von Project Jigsaw .....	827
15.2	Modularisierung im Überblick .....	828
15.2.1	Grundlagen zu Project Jigsaw .....	828
15.2.2	Beispiel mit zwei Modulen .....	834
15.2.3	Packaging .....	843
15.2.4	Abhängigkeiten und Modulgraphen .....	845
15.2.5	Module des JDKs einbinden .....	847
15.2.6	Arten von Modulen .....	853
15.3	Sichtbarkeiten und Zugriffsschutz .....	854
15.3.1	Sichtbarkeiten .....	854
15.3.2	Zugriffsschutz und Reflection .....	856
15.4	Empfehlenswertes Verzeichnislayout für Module .....	858
15.5	Kompatibilität und Migration .....	860
15.5.1	Kompatibilitätsmodus .....	860
15.5.2	Migrationsszenarien .....	863
15.5.3	Fallstrick bei der Bottom-up-Migration .....	867
15.5.4	Beispiel: Migration mit Automatic Modules .....	868
15.5.5	Beispiel: Automatic und Unnamed Module .....	870
15.5.6	Beispiel: Abwandlung mit zwei Automatic Modules .....	873
15.5.7	Fazit .....	875
15.6	Zusammenfassung .....	876

**V Fallstricke und Lösungen im Praxisalltag 877**

<b>16</b>	<b>Bad Smells</b> .....	<b>879</b>
16.1	Programmdesign .....	881
16.1.1	Bad Smell: Verwenden von Magic Numbers .....	881
16.1.2	Bad Smell: Konstanten in Interfaces definieren .....	882
16.1.3	Bad Smell: Zusammengehörende Konstanten nicht als Typ definiert .....	884
16.1.4	Bad Smell: Casts auf unbekannte Subtypen .....	886
16.1.5	Bad Smell: Programmcode im Logging-Code .....	888
16.1.6	Bad Smell: Dominanter Logging-Code .....	889
16.1.7	Bad Smell: Unvollständige Änderungen nach Copy-Paste ..	890
16.1.8	Bad Smell: Unvollständige Betrachtung aller Alternativen ..	892
16.1.9	Bad Smell: Prä-/Postinkrement in komplexeren Statements	893
16.1.10	Bad Smell: Mehrere aufeinanderfolgende Parameter gleichen Typs .....	896
16.1.11	Bad Smell: Grundloser Einsatz von Reflection .....	897
16.1.12	Bad Smell: <code>System.exit()</code> mitten im Programm .....	898
16.1.13	Bad Smell: Variablendeklaration nicht im kleinstmöglichen Sichtbarkeitsbereich .....	900
16.2	Klassendesign .....	902
16.2.1	Bad Smell: Unnötigerweise veränderliche Attribute .....	902
16.2.2	Bad Smell: Herausgabe von <code>this</code> im Konstruktor .....	904
16.2.3	Bad Smell: Aufruf abstrakter Methoden im Konstruktor ...	905
16.2.4	Bad Smell: Mix abstrakter und konkreter Basisklassen .....	910
16.2.5	Bad Smell: Referenzierung von Subklassen in Basisklassen	912
16.2.6	Bad Smell: Öffentlicher Defaultkonstruktor lediglich zum Zugriff auf Hilfsmethoden .....	914
16.3	Fehlerbehandlung und Exception Handling .....	915
16.3.1	Bad Smell: Unbehandelte Exception .....	915
16.3.2	Bad Smell: Unpassender Exception-Typ .....	917
16.3.3	Bad Smell: Fangen der allgemeinsten Exception .....	918
16.3.4	Bad Smell: Exceptions zur Steuerung des Kontrollflusses ..	920
16.3.5	Bad Smell: Unbedachte Rückgabe von <code>null</code> .....	922
16.3.6	Bad Smell: Rückgabe von <code>null</code> statt Exception im Fehlerfall	924
16.3.7	Bad Smell: Sonderbehandlung von Randfällen .....	926
16.3.8	Bad Smell: Keine Gültigkeitsprüfung von Eingabeparametern	927
16.3.9	Bad Smell: Fehlerhafte Fehlerbehandlung .....	929
16.3.10	Bad Smell: I/O ohne <code>finally</code> oder ARM .....	931
16.3.11	Bad Smell: Resource Leaks durch Exceptions im Konstruktor	932
16.4	Häufige Fallstricke .....	937
16.5	Weiterführende Literatur .....	952

<b>17</b>	<b>Refactorings</b> .....	<b>953</b>
17.1	Refactorings am Beispiel .....	954
17.2	Das Standardvorgehen .....	962
17.3	Kombination von Basis-Refactorings .....	965
17.3.1	Refactoring-Beispiel: Ausgangslage und Ziel .....	965
17.3.2	Auflösen der Abhängigkeiten .....	967
17.3.3	Vereinfachungen .....	974
17.3.4	Verlagern von Funktionalität .....	978
17.4	Der Refactoring-Katalog .....	979
17.4.1	Reduziere die Sichtbarkeit von Attributen .....	979
17.4.2	Minimiere veränderliche Attribute .....	982
17.4.3	Reduziere die Sichtbarkeit von Methoden .....	986
17.4.4	Ersetze Mutator- durch Business-Methode .....	988
17.4.5	Minimiere Zustandsänderungen .....	989
17.4.6	Führe ein Interface ein .....	989
17.4.7	Spalte ein Interface auf .....	990
17.4.8	Führe ein Read-only-Interface ein .....	991
17.4.9	Führe ein Read-Write-Interface ein .....	992
17.4.10	Lagere Funktionalität in Hilfsmethoden aus .....	993
17.4.11	Trenne Informationsbeschaffung und -verarbeitung .....	995
17.4.12	Wandle Konstantensammlung in <code>enum</code> um .....	1002
17.4.13	Entferne Exceptions zur Steuerung des Kontrollflusses ....	1004
17.4.14	Wandle in Utility-Klasse mit statischen Hilfsmethoden um ..	1007
17.4.15	Löse <code>if-else</code> / <code>instanceof</code> durch Polymorphie auf ....	1010
17.5	Defensives Programmieren .....	1013
17.5.1	Führe eine Zustandsprüfung ein .....	1013
17.5.2	Überprüfe Eingabeparameter .....	1014
17.6	Fallstricke bei Refactorings .....	1019
17.7	Weiterführende Literatur .....	1021
<b>18</b>	<b>Entwurfsmuster</b> .....	<b>1023</b>
18.1	Erzeugungsmuster .....	1026
18.1.1	Erzeugungsmethode .....	1026
18.1.2	Fabrikmethode (Factory Method) .....	1029
18.1.3	Erbauer (Builder) .....	1032
18.1.4	Singleton .....	1035
18.1.5	Prototyp (Prototype) .....	1040
18.2	Strukturmuster .....	1044
18.2.1	Fassade (Façade) .....	1044
18.2.2	Adapter .....	1047
18.2.3	Dekorierer (Decorator) .....	1049
18.2.4	Kompositum (Composite) .....	1052

18.3	Verhaltensmuster .....	1056
18.3.1	Iterator .....	1056
18.3.2	Null-Objekt (Null Object) .....	1058
18.3.3	Schablonenmethode (Template Method) .....	1061
18.3.4	Strategie (Strategy) .....	1065
18.3.5	Befehl (Command) .....	1077
18.3.6	Proxy .....	1084
18.3.7	Beobachter (Observer) .....	1086
18.3.8	MVC-Architektur .....	1095
18.4	Weiterführende Literatur .....	1097

## **VI Qualitätssicherungsmaßnahmen 1099**

<b>19</b>	<b>Programmierstil und Coding Conventions .....</b>	<b>1101</b>
19.1	Grundregeln eines guten Programmierstils .....	1101
19.1.1	Keep It Human-Readable .....	1102
19.1.2	Keep It Simple And Short (KISS) .....	1102
19.1.3	Keep It Natural .....	1102
19.1.4	Keep It Clean .....	1103
19.2	Die Psychologie beim Sourcecode-Layout .....	1103
19.2.1	Gesetz der Ähnlichkeit .....	1103
19.2.2	Gesetz der Nähe .....	1105
19.3	Coding Conventions .....	1106
19.3.1	Grundlegende Namens- und Formatierungsregeln .....	1107
19.3.2	Namensgebung .....	1110
19.3.3	Dokumentation .....	1113
19.3.4	Programmdesign .....	1115
19.3.5	Klassendesign .....	1120
19.3.6	Parameterlisten .....	1124
19.3.7	Logik und Kontrollfluss .....	1125
19.4	Sourcecode-Prüfung mit Tools .....	1128
19.4.1	Metriken .....	1129
19.4.2	Sourcecode-Prüfung im Build-Prozess .....	1133
<b>20</b>	<b>Unit Tests .....</b>	<b>1143</b>
20.1	Testen im Überblick .....	1143
20.1.1	Was versteht man unter Testen? .....	1143
20.1.2	Testarten im Überblick .....	1144
20.1.3	Zuständigkeiten beim Testen .....	1147
20.1.4	Testen und Qualität .....	1150
20.2	Wissenswertes zu Testfällen .....	1154
20.2.1	Testfälle mit JUnit definieren .....	1154

20.2.2	Problem der Kombinatorik beim Bestimmen von Testfällen .	1163
20.3	Besondere Assertions und Annotations . . . . .	1171
20.4	Parametrisierte Tests mit JUnit 5 . . . . .	1182
20.4.1	Einstieg . . . . .	1182
20.4.2	Verbesserung des Tests der Rabattberechnung . . . . .	1189
20.4.3	Praxisbeispiel: Berechnung in Testfall vereinfachen . . . . .	1191
20.4.4	Praxis-Trickkiste . . . . .	1194
20.5	Fortgeschrittene Unit-Test-Techniken . . . . .	1201
20.5.1	Stellvertreterobjekte / Test-Doubles . . . . .	1204
20.5.2	Vorarbeiten für das Testen mit Stubs und Mocks . . . . .	1208
20.5.3	Die Technik EXTRACT AND OVERRIDE . . . . .	1210
20.5.4	Einstieg in das Testen mit Mocks und Mockito . . . . .	1217
20.5.5	Abhängigkeiten mit Mockito auflösen . . . . .	1226
20.5.6	Unit Tests von privaten Methoden . . . . .	1228
20.6	Test Smells . . . . .	1230
20.6.1	Test Smell: Unangebrachtes <code>assertTrue()</code> und <code>assertFalse()</code> . . . . .	1230
20.6.2	Test Smell: Zu viele Asserts im Testfall . . . . .	1231
20.6.3	Test Smell: Asserts ohne Hinweis . . . . .	1233
20.6.4	Test Smell: Einsatz von <code>toString()</code> in <code>assertEquals()</code> . . . . .	1234
20.6.5	Test Smell: Unit Tests zur Prüfung von Laufzeiten . . . . .	1235
20.7	Nützliche Tools für Unit Tests . . . . .	1237
20.7.1	Hamcrest . . . . .	1237
20.7.2	AssertJ . . . . .	1241
20.7.3	MoreUnit . . . . .	1246
20.7.4	Infinittest . . . . .	1247
20.7.5	JaCoCo . . . . .	1248
20.7.6	EclEmma . . . . .	1252
20.8	Umstieg von JUnit 4 auf JUnit 5 . . . . .	1254
20.8.1	Erweiterung im Gradle-Build für JUnit-5-Tests . . . . .	1254
20.8.2	Veränderungen in den Annotations . . . . .	1254
20.8.3	Alternativen für JUnit Rules . . . . .	1256
20.8.4	Veränderungen bei parametrisierten Tests . . . . .	1260
20.8.5	Alternative zur Hamcrest-Integration in JUnit 4 . . . . .	1261
20.9	Fazit . . . . .	1262
20.10	Weiterführende Literatur . . . . .	1263
<b>21</b>	<b>Codereviews . . . . .</b>	<b>1265</b>
21.1	Definition . . . . .	1265
21.2	Probleme und Tipps zur Durchführung . . . . .	1267
21.3	Vorteile von Codereviews . . . . .	1269
21.4	Codereview-Checkliste . . . . .	1272

<b>22</b>	<b>Optimierungen</b> .....	<b>1273</b>
22.1	Grundlagen .....	1274
22.1.1	Optimierungsebenen und Einflussfaktoren .....	1275
22.1.2	Optimierungstechniken .....	1276
22.1.3	CPU-bound-Optimierungsebenen am Beispiel .....	1278
22.1.4	Messungen – Erkennen kritischer Bereiche .....	1282
22.1.5	Abschätzungen mit der O-Notation .....	1289
22.2	Einsatz geeigneter Datenstrukturen .....	1292
22.2.1	Einfluss von Arrays und Listen .....	1293
22.2.2	Optimierungen für <code>Set</code> und <code>Map</code> .....	1297
22.2.3	Design eines Zugriffsinterface .....	1299
22.3	Lazy Initialization .....	1302
22.3.1	Konsequenzen des Einsatzes der Lazy Initialization .....	1305
22.3.2	Lazy Initialization mithilfe des <code>PROXY</code> -Musters .....	1307
22.4	Optimierungen am Beispiel .....	1310
22.5	I/O-bound-Optimierungen .....	1318
22.5.1	Technik – Wahl passender Strategien .....	1318
22.5.2	Technik – Caching und Pooling .....	1321
22.5.3	Technik – Vermeidung unnötiger Aktionen .....	1321
22.6	Memory-bound-Optimierungen .....	1325
22.6.1	Technik – Wahl passender Strategien .....	1325
22.6.2	Technik – Caching und Pooling .....	1326
22.6.3	Optimierungen der Stringverarbeitung .....	1333
22.6.4	Technik – Vermeidung unnötiger Aktionen .....	1336
22.7	CPU-bound-Optimierungen .....	1338
22.7.1	Technik – Wahl passender Strategien .....	1339
22.7.2	Technik – Caching und Pooling .....	1342
22.7.3	Technik – Vermeidung unnötiger Aktionen .....	1344
22.8	Weiterführende Literatur .....	1347
<b>23</b>	<b>Schlussgedanken</b> .....	<b>1349</b>

<b>VII</b>	<b>Anhang</b>	<b>1351</b>
------------	---------------	-------------

<b>A</b>	<b>Grundlagen zur Java Virtual Machine</b> .....	<b>1353</b>
A.1	Wissenswertes rund um die Java Virtual Machine .....	1353
A.1.1	Ausführung eines Java-Programms .....	1353
A.1.2	Speicherverwaltung und Classloading .....	1354
	<b>Literaturverzeichnis</b> .....	<b>1357</b>
	<b>Index</b> .....	<b>1361</b>

# Vorwort



# Vorwort zur 5. Auflage

Sie halten die mittlerweile 5. Auflage dieses Buchs in den Händen. Das wurde nur durch den großen Zuspruch und das auch nach Jahren anhaltende Interesse für dieses Buch möglich. Somit geht zunächst ein herzlicher Dank an alle Leser der vorherigen Auflagen.

Diese 5. Auflage wurde vollständig auf Java 11 als derzeitige LTS-Version (Long Term Support) aktualisiert sowie in diversen Teilen überarbeitet und erweitert. Dieses Buch soll Ihnen einen fundierten Einstieg in die professionelle Java-Programmierung ermöglichen und damit Ihren Weg zum Java-Profi erleichtern. Wie schon aus den Vorgängern gewohnt, gebe ich immer wieder Tipps aus dem Praxisalltag, weise auf Fallstricke hin und zeige Lösungswege auf. Damit Sie aber am Puls der Zeit sind und über alles Wesentliche bis hin zum aktuellen Java 15 Bescheid wissen, behandle ich die vielfältigen Neuerungen ebenso wie die Modularisierung in jeweils eigenen Kapiteln. Für eine noch umfassendere Behandlung der Thematik verweise ich Sie auf mein Buch »Java – die Neuerungen in Version 9 bis 14: Modularisierung, Syntax- und API-Erweiterungen« [41].

## Änderungen in dieser 5. Auflage

Im Rahmen der Überarbeitung für diese 5. Auflage habe ich das Buch nochmals vollständig gelesen und kritisch beleuchtet. Dadurch konnten kleinere Unstimmigkeiten, missverständliche Formulierungen und ein paar verbliebene Tippfehler erkannt und korrigiert werden. Zudem habe ich die Anregungen und Wünsche von Lesern sowie von Kollegen und Freunden mit eigenen Ideen kombiniert. Daraus sind diverse Ergänzungen und Überarbeitungen in den bereits vorhandenen Kapiteln entstanden. Auch wurden verschiedene Dinge restrukturiert und thematisch neu gegliedert.

Nachfolgend liste ich wesentliche Änderungen dieser 5. Auflage im Vergleich zum Vorgänger auf:

- Kapitel 2 »**Professionelle Arbeitsumgebung**« – Der Text wurde leicht überarbeitet und aktualisiert, das gilt etwa für die Einführung ins Unit-Testen. Diese setzt nun auf JUnit 5. Darüber hinaus behandle ich das Build-Tool Ant in dieser 5. Auflage nicht mehr, da es in der Praxis kaum noch eine Rolle spielt. Zudem wurde die Beschreibung von Gradle auf die im September 2020 aktuelle Version 6.6.1 angepasst.

- Kapitel 3 »**OO-Design**« – In diesem Kapitel wurden ein paar Details und Beispiele leicht überarbeitet, um die Verständlichkeit weiter zu verbessern, etwa im Bereich der kovarianten Rückgabewerte. Zudem weise ich, wo sinnvoll, auf mögliche Vereinfachungen durch aktuelle Java-14- bzw. Java-15-Sprachmittel hin.
- Kapitel 4 »**Lambdas, Methodenreferenzen und Defaultmethoden**« – Dieses Kapitel enthält nun Neuerungen für Interfaces in Java 9. Außerdem wurde eine Beschreibung zu einer kosmetischen Erweiterung in `Predicate<T>` in JDK 11 ergänzt. Schließlich thematisiere ich nun ein kleines, aber wichtiges Detail bei der Definition von Lambdas und dem Zugriff auf Variablen.
- Kapitel 5 »**Java-Grundlagen**« – Die Behandlung der Java-Grundlagen wurde gestrafft, etwa im Bereich des alten APIs zur Dateiverwaltung. Zudem habe ich ganze Abschnitte wie denjenigen zum alten Datums-API entfernt und diverse Beispiele auf das moderne Date and Time API umgestellt. Außerdem wurden wesentliche Neuerungen aus Java 9 bis 11 an passenden Stellen, etwa im Bereich für Strings und auch für die Utility-Klasse `Files` oder die Syntaxneuerung `var`, integriert.
- Kapitel 6 »**Das Collections-Framework**« – Neben Detailkorrekturen inklusive inhaltlicher Straffung wurden einige Neuerungen aus Java 8 und 9 an passender Stelle hinzugefügt, insbesondere Collection-Factory-Methoden sowie Hilfsmethoden in der Klasse `Arrays`. Die Beschreibung zur Klasse `Optional<T>` wurde überarbeitet und um die Neuerungen aus Java 10 und 11 erweitert.
- Kapitel 7 »**Stream-API**« – In diesem Kapitel wurden einige inhaltliche Ergänzungen, etwa für die Kombination von `skip()` und `limit()`, sowie kleinere sprachliche Korrekturen vorgenommen. Darüber hinaus ist vor allem ein Abschnitt zu den Neuerungen im Stream-API in Java 9 hinzugefügt worden. Die Verarbeitung von ZIP-Dateien wurde als Praxisbeispiel entfernt.
- Kapitel 8 »**Datumsverarbeitung seit JDK 8**« – Dieses Kapitel hat ebenfalls kleine sprachliche Korrekturen sowie minimale Anpassungen zur Stringenz der Beispiele erfahren. Zudem werden nun Erweiterungen aus Java 9 beschrieben. Um die Möglichkeiten der Datumsarithmetik noch besser nachvollziehen zu können, habe ich zwei Beispiele ergänzt. Besonders erwähnenswert ist dasjenige zur Berechnung des Gehaltszahltags als Real-World-Example. Dieses verfolgt ein schrittweises Vorgehen mit dem Hinzufügen von jeweils kleinen Funktionsblöcken und zeigt auch gleich geeignetes Testing mit JUnit 5.
- Kapitel 9 »**Applikationsbausteine**« – In diesem Kapitel wurden diverse Kleinigkeiten angepasst und Kürzungen vorgenommen. Das betrifft vor allem die Themen Preferences sowie Wertebereichsprüfungen. Bei Letzterem wurde auf die Darstellung eigener Lösungen zugunsten von Google Guava verzichtet. Zudem habe ich das abschließende Beispiel zur Verarbeitung von CSV so umgestaltet, dass es nun das Java-14-Feature `records` nutzt.

- Kapitel 10 »**Multithreading**« – Dieses Kapitel hat diverse Detailänderungen erfahren und der Teil zu Producer-Consumer wurde gekürzt. Weil mittlerweile modernere und zu bevorzugende Konstrukte existieren, habe ich die Beschreibung zur zeitgesteuerten Ausführung mit `Timer` und `TimerTask` entfernt. Insbesondere wurden fortgeschrittenere Themen und die Beschreibung moderner APIs in ein eigenständiges Kapitel ausgelagert.
- Kapitel 11 »**Modern Concurrency**« – Dieses Kapitel wurde komplett neu gestaltet und diverse Beispiele ergänzt und prägnanter formuliert. Ziel ist es, modernere Concurrency-Konzepte vorzustellen. Die hier beschriebenen Concurrency Utilities und das Executor- sowie das Fork-Join-Framework erlauben es, statt auf Low-Level-Ebene zu parallelisierende Abläufe auf konzeptionell höherer Ebene umzusetzen. Aber es wird noch besser: Verarbeitungsabläufe lassen sich mithilfe von `CompletableFuture` sehr elegant beschreiben und parallelisieren. Abschließend thematisiere ich die Reactive Streams, die eine Verbindung zum Reactive Programming bilden.
- Kapitel 12 »**Fortgeschrittene Java-Themen**« – Der Text hat ein Facelift erhalten. Die Beispiele wurden durchgehend auf das Date and Time API aktualisiert. Insbesondere habe ich den Teil zur Garbage Collection deutlich gekürzt. Vollständig entfernt wurde die Beschreibung zur JavaScript-Verarbeitung, da diese mit Java 15 nicht mehr Bestandteil des JDKs ist. Eine Beschreibung zum HTTP/2-API wurde in diesem Kapitel ergänzt. Als Schmankerl schauen wir uns dort an, wie man damit REST-Calls absetzen und auf diese Weise Wechselkurse ermitteln kann.
- Kapitel 13 »**Basiswissen Internationalisierung**« – Der Text wurde in den Bereichen gekürzt, die das alte Datums-API behandelt haben. Diverse Teile verwenden jetzt das modernere Date and Time API aus Java 8. Da Java im Bereich Desktop und GUIs kaum mehr verwendet wird, habe ich die Beispiele, die JavaFX oder Swing nutzen, entfernt.
- Kapitel 17 »**Refactorings**« – Auch das Kapitel zu Refactorings hat einen Facelift erfahren. Vor allem wurden alle begleitenden Unit Tests auf JUnit 5 umgestellt. Zudem kam in einem Beispiel noch die Bibliothek Joda-Time zum Einsatz. Das wurde nun auf das Date and Time API des JDKs umgestellt.
- Kapitel 19 »**Programmierstil und Coding Conventions**« – Dieses Kapitel enthält Detailkorrekturen. Dabei wurden vor allem die Abschnitte zu den Tools überarbeitet und aktualisiert. Die Beschreibung zum Tool FindBugs sowie dessen Nachfolger SpotBugs wurde entfernt, da diese keine neuen Java-Versionen unterstützen, derzeit nicht einmal Java 10 und 11.

- Kapitel 20 »**Unit Tests**« – Auch hier wurde der gesamte Text überarbeitet und umstrukturiert. Vor allem wird nun konsequent das aktuelle JUnit 5 genutzt, wodurch sich einige Testfälle deutlich leichter als mit den Vorgängern beschreiben lassen. Insbesondere den parametrisierten Tests wird viel Aufmerksamkeit gewidmet. Test Smells werden nun ausführlicher behandelt und zeigen konkrete Lösungsmöglichkeiten.
- Kapitel 22 »**Optimierungen**« – Die Performance-Messungen wurden mit JDK 14 wiederholt. Einige Messungen erfolgten ergänzend mit dem seit JDK 12 ins JDK integrierten JMH (Java Microbenchmark Harness). Beim Kopieren von Dateiinhalten wird nun auch die Variante mit `transferTo()` als Neuerung aus Java 9 betrachtet. Bei den CPU-bound-Optimierungen zeige ich mit Memoization eine sehr effiziente Technik, um rekursive Berechnungen dramatisch beschleunigen zu können.

### Erweiterungen

- Kapitel 14 »**Ergänzungen in Java 12 bis 15**« – In den letzten 3 Jahren sind diverse Java-Versionen erschienen, die verschiedene interessante Neuerungen in der Syntax und den APIs mit sich bringen. Wichtige Themen werden in diesem Kapitel vorgestellt.

### Entfallene Themen

Aus drucktechnischen Gründen mussten die Kapitel zu JavaFX sowie zur GUI-Programmierung mit Swing aus der Druckvariante des Buchs entfernt werden. Diese stehen aber – wie auch einige in früheren Auflagen entfernte Anhänge zu UML und dem Softwareentwicklungsprozess – als PDF auf der Seite des Verlags zum Download bereit.

## Danksagung

Ein Fachbuch zu schreiben ist eine schöne, aber arbeitsreiche und langwierige Aufgabe. Alleine kann man dies kaum bewältigen, daher möchte ich mich an dieser Stelle bei allen bedanken, die direkt oder indirekt dazu beigetragen haben.

Bei der Erstellung des Manuskripts konnte ich auf ein starkes Team an Korrekturlesern zurückgreifen, insbesondere wieder einmal auf Michael Kulla, der akribisch gelesen und viele hilfreiche Anmerkungen eingebracht hat. Vielen Dank! Für das neu erstellte Kapitel Modern Concurrency haben Prof. Dr. Rainer Oechsle und René Preißel diverse nützliche Anmerkungen zur Verbesserung beigesteuert und kleinere Unstimmigkeiten aufgedeckt. In diesem Kontext erhielt ich ein paar hilfreiche Anmerkungen von Dr. Heinz Kabutz. Besten Dank an alle!

Wie immer geht natürlich auch ein Dankeschön an das Team des dpunkt.verlags (vor allem Dr. Michael Barabas, Anja Weimer und Stefanie Weidner) für die gute 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 natürlich ein lieber Dank an meine Frau Lilija für ihr Verständnis und ihre Unterstützung. Bei der Erstellung dieser 5. Auflage war ich glücklicherweise weit weniger im Stress als bei den vorherigen Ausgaben, sodass sogar die eine oder andere gemeinsame Fahrradtour oder Wanderung möglich wurde. Auch die Vorbereitungen auf die Geburt unserer Tochter laufen auf Hochtouren und die Freude ist groß.

## Anregungen und Kritik

Ich wünsche allen Lesern viel Freude und einige neue Erkenntnisse durch die Lektüre dieser 5. Auflage. Möge Ihnen der »Weg zum Java-Profi« mit meinem Buch ein wenig leichter fallen.

Trotz großer Sorgfalt lassen sich Fehler bei einem so umfangreichen Buch leider nicht vollständig vermeiden. Falls Ihnen ein solcher auffällt oder eine Formulierung missverständlich sein sollte, so zögern Sie bitte nicht, mir dies mitzuteilen. Haben Sie Anregungen, Verbesserungsvorschläge oder fehlt Ihnen noch eine Information? Sie erreichen mich per Mail unter: [michael\\_inden@hotmail.com](mailto:michael_inden@hotmail.com).

Zürich, im Oktober 2020

Michael Inden

## Danksagung zur 4. Auflage

Bei der Erstellung des Manuskripts konnte ich auf ein starkes Team an Korrekturlesern zurückgreifen, insbesondere diesmal auch auf Michael Kulla, der akribisch gelesen und viele hilfreiche Anmerkungen eingebracht hat. Vielen Dank! Wieder einmal haben mich meine Freunde Merten Driemeyer, Dr. Clemens Gugenberger, Prof. Dr. Carsten Kern und Andreas Schöneck hervorragend unterstützt. Den einen oder anderen Hinweis und Tipp erhielt ich von Jeton Memeti, Marius Reusch und Prof. Dr. Andreas Spillner.

Weil der Java-9-Teil inhaltlich – wenn auch vom Umfang deutlich abgespeckt – weitestgehend meinem Buch »Java 9 – Die Neuerungen« [40] entstammt, danke ich allen dort Beteiligten ebenfalls.

## Danksagung zur 3. Auflage

Bei der Erstellung des Manuskripts konnte ich auf ein starkes Team an Korrekturlesern zurückgreifen, insbesondere diesmal auch auf Benjamin Muschko und Hans Dockter als Experten zu Gradle sowie Hendrik Schreiber, selbst Autor eines Java-Fachbuchs zu Optimierungen. Vielen Dank an euch!

Einige Tipps erhielt ich von Tim Bötzmeyer und Reinhard Pupkes. Auch haben mich folgende Personen hervorragend unterstützt: Merten Driemeyer, Dr. Clemens Gugenberger, Dr. Carsten Kern, Florian Messerschmidt und Andreas Schöneck. Darüber hinaus kamen gute Anmerkungen von verschiedenen Zühlke-Kollegen: Michael Haspra, Jörg Keller, Rick Janda, Franziska Meyer, Sagi Nedunkanal, Joachim Prinzbach und Dr. Christoph Schmitz. Der Java-8-Teil entstammt weitestgehend meinem Buch »Java 8 – Die Neuerungen«. Allen dort Beteiligten danke ich ebenfalls.

Neben den Korrekturlesern möchte ich einen ganz herzlichen Dank an meinen damaligen Arbeitgeber Zühlke Engineering AG und insbesondere meinen Chef Wolfgang Giersche für die gewährte freie Zeit zum Finalisieren des Buchs aussprechen. Das war eine große Hilfe in der letzten heißen Phase vor der Abgabe des Manuskripts.

## Danksagung zur 2. Auflage

Bei der Erstellung der 2. Auflage konnte ich wieder auf ein starkes Team an Korrekturlesern zurückgreifen. Einige Tipps erhielt ich von Dr. Alexander Kort und Reinhard Pupkes. Auch haben mich folgende Personen hervorragend unterstützt: Stefan Bartels, Tim Bötzmeyer, Rudolf Braun, Andreas Bubolz, Merten Driemeyer, Bernd Eckstein, Dr. Clemens Gugenberger, Peter Kehren, Dr. Carsten Kern, Dr. Iris Rottländer, Roland Schmitt-Hartmann und Andreas Schöneck.

Dabei möchte ich folgende vier Personen herausheben: Stefan Bartels für seine sprachliche Gründlichkeit, Andreas Bubolz für seine Genauigkeit und Dr. Clemens Gugenberger sowie Andreas Schöneck für die ganzen hilfreichen Anregungen.

## Danksagung zur 1. Auflage

Zu meiner Zeit bei der Heidelberger Druckmaschinen AG in Kiel ist bei den Vorbereitungen zu Codereviews und der Ausarbeitung von Vorträgen zum ersten Mal der Gedanke an ein solches Buch entstanden. Danke an meine damaligen Kollegen, die an diesen Meetings teilgenommen haben. Als Veranstalter und Vortragender lernt man immer wieder neue Details. Dietrich Mucha und Reinhard Pupkes danke ich für ihre Korrekturen und Anmerkungen, die gemeinsamen Erfahrungen beim Ausarbeiten von Coding Conventions und Codereviews sowie die nette Zeit beim Pair Programming. Die Zusammenarbeit mit Tim Bötzmeyer hat mir viel Freude bereitet. Unsere langen, interessanten Diskussionen über Java und die Fallstricke beim OO-Design haben mir diverse neue Einblicke verschafft.

Auch einige Kollegen bei der IVU Traffic Technologies AG in Aachen haben mich mit Korrekturen und Anregungen unterstützt. Unter anderem danke ich Rudolf Braun, Christian Gehrman, Peter Kehren, Felix Korb und Roland Schmitt-Hartmann für den einen oder anderen Hinweis und Tipp, um den Text weiter zu verbessern. Mein spezieller Dank gilt Merten Driemeyer, der sich sehr gründlich mit frühen Entwürfen des Manuskripts beschäftigt und mir an diversen Stellen fachliche und sprachliche Tipps gegeben hat. Gleiches gilt für Dr. Iris Rottländer, die sowohl formal als auch inhaltlich an vielen Stellen durch ihre Anmerkungen für eine Verbesserung des Textes gesorgt hat. Auch Dr. Carsten Kern und Andreas Schöneck haben gute Hinweise gegeben und einige verbliebene kleinere Fehler aufgedeckt. Last, but not least haben die Anmerkungen von Dr. Clemens Gugenberger und unsere nachfolgenden Diskussionen einigen Kapiteln den letzten Feinschliff gegeben. Gleiches gilt für Stefan Bartels, der mich immer wieder durch gute Anmerkungen unterstützt und damit zur Verständlichkeit des Textes beigetragen hat. Alle sechs haben mir entscheidend geholfen, inhaltlich für mehr Stringenz und Klarheit zu sorgen. Mein größter Dank geht an Andreas Bubolz, der mich immer wieder unterstützt und enorm viel Zeit und Mühe investiert hat. Als Korrekturleser und Sparringspartner in vielen Diskussionen hat er diverse Unstimmigkeiten im entstehenden Text aufgedeckt.

