

Inhaltsverzeichnis

1	Einleitung	1
1.1	Releasepolitik	1
1.2	Inhaltsübersicht	4
1.2.1	Neuerungen in Java 12 bis 17 LTS	4
1.2.2	Neuerungen in Java 18 bis 21 LTS	5
1.3	Grundgerüst des Beispielprojekts	6
1.4	Anmerkung zum Programmierstil	7
1.4.1	Gedanken zur Sourcecode-Kompaktheit	7
1.4.2	Gedanken zu <code>final</code> und <code>var</code>	8
1.4.3	Blockkommentare in Listings	9
1.4.4	Gedanken zur Formatierung	9
1.5	Installation von Java 21 LTS	10
1.5.1	Java-Download	10
1.5.2	Installation des JDKs	10
1.5.3	Installationsnachteile	11
1.5.4	Java-Installation prüfen	13
1.6	Konfigurationen für Build-Tools und IDEs	14
1.6.1	Java 21 LTS mit Gradle	14
1.6.2	Java 21 LTS mit Maven	15
1.6.3	Java 21 LTS mit IntelliJ	16
1.6.4	Java 21 LTS mit Eclipse	17
1.7	Ausprobieren der Beispiele und Lösungen	19
1.7.1	Ausprobieren neuer Java-Features mit der JShell oder der Kommandozeile	19
1.7.2	Ausprobieren neuer Java-Features in einer Sandbox	22

I Neuerungen in Java 12 bis 17 LTS	23
2 Syntaxneuerungen in JDK 12 bis 17 LTS	25
2.1 Text Blocks	26
2.1.1 Grundlegende Syntax	26
2.1.2 Besonderheiten und Annehmlichkeiten	27
2.1.3 Escaping	30
2.1.4 Platzhalter	32
2.2 Switch Expressions	34
2.2.1 Einführendes Beispiel	34
2.2.2 Vollständigkeitsprüfung	36
2.2.3 Fallstricke der alten Syntax und Abhilfen	37
2.2.4 Rückgabe mit <code>yield</code>	39
2.2.5 Rückwärtskompatible Angabe bei <code>case</code> mit <code>yield</code>	39
2.3 Records	41
2.3.1 Einführendes Beispiel	42
2.3.2 Records für DTOs und Rückgabewerte	44
2.3.3 Erweiterungsmöglichkeiten	46
2.3.4 Besonderheit – Generics in Records	50
2.3.5 Besonderheit – Records und Interfaces	52
2.3.6 Zusammenfassung	54
2.4 Records – Beyond the basics	55
2.4.1 Einfluss der Datenstruktur – Speicherung in verschiedenen Typen von Sets	55
2.4.2 Records und deren Auswirkung auf Builder	58
2.4.3 Records und Immutability	61
2.4.4 Lokale Records (als innere Typen zur temporären Datenaggregation)	65
2.5 Pattern Matching bei <code>instanceof</code>	67
2.5.1 Neue Syntax und einführendes Beispiel	68
2.5.2 Im Praxiseinsatz	69
2.6 Sealed Types	71
2.6.1 Einführendes Beispiel	71
2.6.2 Praxisbeispiel mit Extension Point	74
2.6.3 Wissenswertes	76
2.7 Lokale Enums und Interfaces	77
2.8 Statische Attribute und Methoden in inneren Klassen	78
2.9 Pattern Matching und Erweiterung für <code>switch</code> (Preview)	79
2.9.1 Einfaches Pattern Matching	79
2.9.2 Pattern Matching mit Bedingung	80
2.9.3 Spezialfall: Behandlung von <code>null</code>	82

3	Übungen zu den Syntaxneuerungen in JDK 12 bis 17 LTS	83
3.1	Aufgaben	83
3.2	Lösungen	89
4	API-Erweiterungen in JDK 12 bis 17 LTS	101
4.1	Erweiterungen in der Klasse <code>String</code>	101
4.1.1	Die Methode <code>indent()</code>	101
4.1.2	Die Methode <code>transform()</code>	104
4.1.3	Die Methode <code>formatted()</code>	105
4.2	Erweiterungen im Stream-API	107
4.2.1	Der <code>teeing()</code> -Kollektor	107
4.2.2	Die Methode <code>toList()</code> als Shortcut zum Kollektor	110
4.2.3	Die Intermediate Operation <code>mapMulti()</code>	111
4.3	Die Utility-Klasse <code>CompactNumberFormat</code>	114
4.3.1	Einführendes Beispiel	114
4.3.2	Rundung steuern	117
4.3.3	Eigene Einheiten angeben	118
4.4	Unix-Domain-Socket Channels	119
4.5	Verschiedenes	123
4.5.1	Die Methode <code>mismatch()</code> in der Utility-Klasse <code>Files</code>	123
4.5.2	JEP 356: Enhanced Pseudo-Random Number Generators	125
4.5.3	Day Period Support	126
4.5.4	Aufruf von Defaultmethoden aus Dynamic Proxies	129
4.6	Deprecations	132
4.6.1	JEP 398: Deprecate the Applet API for Removal	132
4.6.2	JEP 407: Remove RMI Activation	133
4.6.3	JEP 411: Deprecate the Security Manager for Removal	133
4.6.4	JEP 415: Context-Specific Deserialization Filters	133
4.6.5	Deprecations der Konstruktoren der Wrapper-Klassen	133
5	Übungen zu den API-Erweiterungen in JDK 12 bis 17 LTS	135
5.1	Aufgaben	135
5.2	Lösungen	138
6	JVM-Neuerungen in JDK 12 bis 17 LTS	143
6.1	Verbesserung bei <code>NullPointerExceptions</code>	144
6.1.1	Einführende Beispiele	144
6.1.2	Fehlersuche bei komplexen Ausdrücken	146
6.2	Microbenchmarks und JMH	148
6.2.1	Eigene Microbenchmarks und Varianten davon	148
6.2.2	Microbenchmarks mit JMH	151
6.2.3	Weitere Microbenchmarks mit JMH	159
6.2.4	Fallstricke beim Benchmarking mit JMH	162

6.2.5 Projekt zum Experimentieren	166
6.2.6 Fazit zu JMH	167
6.3 Das Packaging-Tool <code>jpacakge</code>	167
6.3.1 Einführung	167
6.3.2 <code>jpacakge</code> am Beispiel	168
6.3.3 Externe Bibliotheken mit <code>jpacakge</code> einbinden	171
6.3.4 Hintergrundwissen: Verzeichnisstruktur und -inhalt der Distributionen	175
6.4 Veränderungen bei den Garbage Collectors	176
6.4.1 Der Garbage Collector namens ZGC	176
6.4.2 Entfernung von Concurrent Mark Sweep (CMS)	176
6.5 JShell und Direct Compilation mit modernem Java	177
6.5.1 JShell: Brandaktuelle und Preview-Features nutzen	177
6.5.2 Direct Compilation: Preview-Features ausprobieren	178
6.6 Verschiedenes	179
6.6.1 JEP 306: Restore Always-Strict Floating-Point Semantics	179
6.6.2 JEP 382: New macOS Rendering Pipeline	179
6.6.3 JEP 391: macOS/AArch64 Port	179
6.6.4 JEP 403: Strongly Encapsulate JDK Internals	179
6.7 Ausgliederungen/Deprecations	180
6.7.1 JEP 410: Remove the Experimental AOT and JIT Compiler	180
6.7.2 Entfernung der JavaScript-Engine	180
7 Übungen zu den JVM-Neuerungen in JDK 12 bis 17 LTS	183
7.1 Aufgaben	183
7.2 Lösungen	184

II Neuerungen in Java 18 bis 21 LTS 187

8 Neuerungen in Java 21 LTS im Überblick	189
8.1 JEPs im Überblick	189
8.1.1 JEPs in Java 18	189
8.1.2 JEPs in Java 19	190
8.1.3 JEPs in Java 20	190
8.1.4 JEPs in Java 21 LTS	190
8.2 Erinnerung: Neue Releasepolitik im Überblick	191
9 Syntaxneuerungen in JDK 18 bis 21 LTS	193
9.1 JEP 430: String Templates (Preview)	194
9.1.1 Einführung	194
9.1.2 String Templates als Alternative	196
9.1.3 String Templates in der Praxis	199

9.1.4 Besonderheiten und alternative String-Prozessoren	202
9.1.5 Eigene Template-Prozessoren	204
9.2 JEP 440: Record Patterns	205
9.2.1 Einführung	205
9.2.2 Verschachtelte Record Patterns	209
9.2.3 Typinferenz bei generischen Record Patterns	214
9.2.4 Zusammenfassung	217
9.3 JEP 441: Pattern Matching for <code>switch</code>	219
9.3.1 Angabe von Bedingungen (Guarded Patterns)	220
9.3.2 Verbesserung der Dominanzprüfung	224
9.3.3 Verbesserung der Vollständigkeitsanalyse	227
9.3.4 Angabe von Record Patterns	229
9.3.5 Qualified Enums	232
9.4 JEP 443: Unnamed Patterns & Variables (Preview)	235
9.4.1 Motivation für unbenannte Variablen und Pattern	235
9.4.2 Abhilfe durch Unnamed Variables and Patterns	237
10 API-Erweiterungen in JDK 18 bis 21 LTS	241
10.1 JEP 416: Reimplement Core Reflection with Method Handles	242
10.2 JEP 418: Internet-Address Resolution SPI	244
10.3 JEP 431: Sequenced Collections	247
10.3.1 Einführende Beispiele	247
10.3.2 Abhilfe Sequenced Collections	248
10.3.3 Blick hinter die Kulissen	250
10.3.4 Sequenced Collections in Action	251
10.4 JEP 444: Virtual Threads	256
10.4.1 Einführende Beispiele	256
10.4.2 Motivation für Virtual Threads in der Praxis	260
10.4.3 Performance-Vergleich für Plattform- und Virtual Threads	262
10.4.4 Fazit	264
10.5 Verschiedenes	265
10.5.1 JEP 400: UTF-8 by Default	265
10.5.2 JEP 452: Key Encapsulation Mechanism API	266
10.5.3 Neuerungen in der Klasse <code>Math</code>	267
10.5.4 Neuerungen beim Erzeugen einer <code>HashMap<></code>	269
10.5.5 Diverse Änderungen	269
10.6 Deprecations	273
10.6.1 JEP 421: Deprecate Finalization for Removal	273
10.6.2 Weitere Deprecations	273

11 JVM-Neuerungen in JDK 18 bis 21 LTS	275
11.1 JEP 408: Simple Web Server	275
11.2 JEP 413: Code Snippets in Java API Documentation	277
11.3 JEP 445: Unnamed Classes and Instance Main Methods (Preview)	278
11.3.1 Einführung	279
11.3.2 Vereinfachung I: Instance <code>main()</code>	280
11.3.3 Vereinfachung II: Unnamed class	280
11.3.4 Weitere Möglichkeiten	281
11.3.5 Mehrere <code>main()</code> – Was passiert im Hintergrund?	282
11.3.6 Java auf dem Weg zur Skript-basierten Ausführung?	282
11.4 Diverses	283
11.4.1 JEP 422: Linux/RISC-V Port	283
11.4.2 JEP 439: Generational ZGC	283
11.4.3 JEP 449: Deprecate the Windows 32-bit x86 Port for Removal	284
11.4.4 JEP 451: Prepare to Disallow the Dynamic Loading of Agents	284
12 Übungen zu den Neuerungen in JDK 18 bis 21 LTS	285
12.1 Aufgaben	285
12.2 Lösungen	290
III Ausblick	297
13 Ausblick auf Java 22	299
13.1 JEP 423: Region Pinning for G1	300
13.2 JEP 447: Statements before <code>super(...)</code> (Preview)	301
13.2.1 Beispiel: Validierung von Basisklassenkonstruktorargumenten	301
13.2.2 Beispiel: Vorbereiten von Basisklassenkonstruktorargumenten	304
13.2.3 Besonderheit: Aktionen vor dem Aufruf von <code>this()</code>	306
13.3 JEP 454: Foreign Function & Memory API	307
13.4 JEP 456: Unnamed Variables & Patterns	311
13.5 JEP 457: Class-File API (Preview)	312
13.6 JEP 458: Launch Multi-File Source-Code Programs	314
13.7 JEP 459: String Templates (Second Preview)	317
13.7.1 Ergänzendes Beispiel	317
13.7.2 Localization	318
13.7.3 Besonderheiten	321
13.8 JEP 460: Vector API (Seventh Incubator)	321
13.8.1 Einführendes Beispiel	322
13.8.2 Weiterführendes Beispiel	324
13.8.3 Auto-Vektorisierung	327
13.9 JEP 461: Stream Gatherers (Preview)	327
13.9.1 Einführung	327

13.9.2 Ausgewählte vordefinierte <code>Gatherer</code> im Überblick	329
13.10 JEP 462: Structured Concurrency (Second Preview)	333
13.10.1 Einführung	333
13.10.2 Umsetzung mit Structured Concurrency	334
13.10.3 Variante mit der Strategie <code>ShutdownOnSuccess</code>	338
13.11 JEP 463: Implicitly Declared Classes	340
13.12 JEP 464: Scoped Values (Second Preview)	342
13.12.1 Grundlagen	343
13.12.2 Weiterführendes Beispiel	345
13.13 Installation von Java 22	349
13.13.1 Weitere Aktionen unter Windows	349
13.13.2 Weitere Aktionen unter macOS	350
13.13.3 Java-22-Installation prüfen	351
13.14 Notwendige Anpassungen für Build-Tools und IDEs (Java 22)	352
13.14.1 Java 22 mit Gradle	352
13.14.2 Java 22 mit Maven	353
13.14.3 Java 22 mit IntelliJ	354
13.14.4 Java 22 mit Eclipse	355
14 Zusammenfassung und Schlusswort	359
14.1 Gedanken zum Umstieg	359
14.2 Fazit	360
14.3 Abschließende Hinweise	362

IV Anhang **363**

A Wesentliches aus Java 8 LTS, 9, 10 und 11 LTS	365
A.1 Einstieg in Lambdas	365
A.1.1 Lambdas am Beispiel	365
A.1.2 Functional Interfaces und SAM-Typen	366
A.1.3 Type Inference und Kurzformen der Syntax	368
A.1.4 Methodenreferenzen	369
A.1.5 Das Interface <code>Predicate<T></code>	370
A.2 Streams im Überblick	372
A.2.1 Streams erzeugen – Create Operations	373
A.2.2 Intermediate und Terminal Operations im Überblick	374
A.2.3 Intermediate Operations	374
A.2.4 Terminal Operations	378
A.3 Neuerungen in der Datumsverarbeitung	379
A.3.1 Die Klassen <code>LocalDate</code> , <code>LocalTime</code> und <code>LocalDateTime</code>	380
A.3.2 Die Klasse <code>Duration</code>	382
A.3.3 Die Klasse <code>Period</code>	383

A.3.4	Datumsarithmetik mit <code>TemporalAdjusters</code>	384
A.4	Diverse Erweiterungen	386
A.4.1	Erweiterungen in der Klasse <code>String</code>	386
A.4.2	Erweiterungen im Interface <code>Comparator<T></code>	388
A.4.3	Die Klasse <code>Optional<T></code>	390
A.4.4	Collection-Factory-Methoden	393
A.4.5	Erweiterungen im NIO und der Klasse <code>Files</code>	394
A.4.6	Die Klasse <code>CompletableFuture<T></code>	397
A.5	Weitere Syntaxneuerungen	400
A.5.1	Syntaxerweiterung <code>var</code>	400
A.5.2	Anonyme innere Klassen und der Diamond Operator	402
A.6	HTTP/2-API	402
A.6.1	Einführendes Beispiel mit dem neuen HTTP/2-API	403
A.6.2	Webseite als Datei herunterladen	404
A.6.3	Asynchrone Verarbeitung	404
A.7	Java + REPL => <code>jshell</code>	406
A.7.1	Einführendes Beispiel	406
A.7.2	Weitere Kommandos und Möglichkeiten	407
A.7.3	Neuere Java-Features nutzen	408
A.7.4	Komplexere Aktionen	409
A.7.5	JShell-API	412
A.8	Launch Single-File Source-Code Programs	414
A.8.1	Einführendes Beispiel	414
A.8.2	Palindrom-Prüfung	415
A.8.3	Besonderheit: Shebang-Skript	415
B	Die Build-Tools Gradle und Maven im Überblick	417
B.1	Projektstruktur für Gradle und Maven	417
B.2	Einführung in Gradle	419
B.2.1	Installation und grundlegende Aktionen	419
B.2.2	Weiterführende Aktionen	427
B.2.3	Fazit	429
B.3	Einführung Maven	430
B.3.1	Maven im Überblick	430
B.3.2	Maven am Beispiel	434
	Literaturverzeichnis	437
	Index	439

Vorwort

Zunächst einmal bedanke ich mich bei Ihnen, dass Sie sich für dieses Buch entschieden haben. Hierin finden Sie eine Vielzahl an Informationen zu den Neuerungen in der aktuellen Java-Version 21 LTS (Long Term Support) sowie alle wesentlichen Neuerungen aus den Vorgängern sowie dem brandaktuellen Java 22.

Zielgruppe

Dies ist kein Buch für Programmierneulinge, sondern richtet sich an Leser, die bereits solides Java-Know-How besitzen und sich kurz und prägnant über die wichtigsten Neuerungen in den Java-Versionen 12 bis 17 LTS sowie 18 bis 21 LTS informieren wollen sowie einen Ausblick auf Java 22 wünschen.

Dieses Buch wendet sich im Speziellen an zwei Zielgruppen:

1. Zum einen sind dies **engagierte Hobbyprogrammierer und Informatikstudenten**, aber auch **Berufseinsteiger**, die Java als Sprache beherrschen und an den Neuerungen in den aktuellen Java-Versionen interessiert sind.
2. Zum anderen ist das Buch für **erfahrene Softwareentwickler und -architekten** gedacht, die ihr Wissen ergänzen oder auffrischen wollen, um für zukünftige Projekte abschätzen zu können, ob und – wenn ja – für welche Anforderungen die neuen Java-Versionen eine gewinnbringende Alternative darstellen können.

Was vermittelt dieses Buch?

Sie als Leser erhalten in diesem Buch neben Theoriewissen eine Vertiefung durch praktische Beispiele, sodass der Umstieg auf Java 17 LTS, das aktuelle Java 21 LTS oder sogar das brandaktuelle Java 22 in eigenen Projekten erfolgreich gemeistert werden kann. Erleichtert wird ein Umstieg durch eine Vielzahl an Übungen zu den wichtigsten Features.

Um die Beispiele des Buchs möglichst präzise und elegant zu halten, verwende ich diverse Features aus Java 8 LTS, 9, 10 und 11 LTS. Deshalb ist es hilfreich, wenn Sie sich damit schon beschäftigt haben. Alle, die eine kleine Auffrischung benötigen, finden zum leichteren Einstieg in Anhang A einen Crashkurs zu den wesentlichen Neuerungen in den Java-Versionen 8 LTS bis 11 LTS.

Aufbau dieses Buchs

Nachfolgend möchte ich die Themen der einzelnen Kapitel kurz vorstellen.

Kapitel 1 – Einleitung Die Einleitung stimmt Sie auf Java 12 bis 21 LTS sowie Java 22 ein und gibt dazu einen Überblick, was Sie so alles in diesem Buch bzw. als Neuerungen erwartet. Zudem thematisiere ich den Programmierstil und zeige auf, welche Voraussetzungen nötig sind, um die aktuellen Java-Versionen mit Build-Tools und IDEs zu verwenden.

Kapitel 2 – Syntaxneuerungen in JDK 12 bis 17 LTS Zunächst widmen wir uns verschiedenen Änderungen an der Syntax von Java. Das Spektrum reicht von mehrzeiligen Strings und sogenanntem Pattern Matching bei `instanceof` bis zu größeren Erweiterungen bei `switch` sowie den Records als eine extrem kompakte Schreibweise zum Deklarieren spezieller Datencontainerklassen. Darüber hinaus gibt es einige für die Praxis oft weniger relevante Dinge, die kurz vorgestellt werden.

Kapitel 3 – Übungen zu den Syntaxneuerungen in JDK 12 bis 17 LTS Dieses Kapitel bietet einen umfangreichen Satz an Übungsaufgaben, um Sie mit den Neuerungen und Änderungen in der Syntax vertraut zu machen. Für sämtliche Aufgaben werden am Kapitelende korrespondierende Musterlösungen präsentiert.

Kapitel 4 – API-Erweiterungen in JDK 12 bis 17 LTS In den APIs des JDKs finden sich diverse Neuerungen und umfasst Ergänzungen in der Klasse `String` sowie im Stream-API, aber auch Neuerungen wie Unix-Domain-Sockets oder die Klasse `CompactNumberFormat`.

Kapitel 5 – Übungen zu den API-Erweiterungen in JDK 12 bis 17 LTS Dieses Kapitel bietet einen umfangreichen Satz an Übungsaufgaben bezüglich der Neuerungen und Änderungen in den APIs. Für sämtliche Aufgaben werden am Kapitelende korrespondierende Musterlösungen präsentiert.

Kapitel 6 – JVM-Neuerungen in JDK 12 bis 17 LTS In diesem Kapitel beschäftigen wir uns mit Änderungen in der JVM, etwa bei der Garbage Collection oder den Möglichkeiten der `jshell` bezogen auf neuere Java-Versionen. Darüber hinaus könnte Sie das Microbenchmark-Framework JMH interessieren. Schließlich behandle ich das Tool `jpackage` zum Erzeugen installierbarer Distributionen.

Kapitel 7 – Übungen zu den JVM-Neuerungen in JDK 12 bis 17 LTS Für einige der Neuerungen in der JVM bietet dieses Kapitel ein paar Übungsaufgaben inklusive einer kurzen Darstellung möglicher Lösungen.

Kapitel 8 – Neuerungen in Java 21 LTS im Überblick Dieses Kapitel listet die Erweiterungen aus Java 21 LTS auf und stellt diese in jeweils eigenen Abschnitten kurz vor. Dadurch haben Sie bereits eine gute Orientierung, in welchem der Folgekapitel Sie dann die Lektüre fortsetzen möchten. Bei generellem Interesse lesen Sie einfach stringent von vorne nach hinten.

Kapitel 9 – Syntaxneuerungen in JDK 18 bis 21 LTS Mit String Templates lässt sich die Konkatenation von variablen und fixen Textbestandteilen vereinfachen. Record Patterns ermöglichen es, Records auf einfache Weise in ihre Bestandteile zerlegen und darauf zugreifen zu können. Das Pattern Matching funktioniert in modernem Java auch in `switch` und wurde funktionell aufgewertet. Schließlich wird es durch Unnamed Variables and Patterns möglich, verschiedene Elemente in einem Record Pattern oder eine Variable durch ein einzelnes `_` als ungenutzt und unbrauchbar zu markieren.

Kapitel 10 – API-Erweiterungen in JDK 18 bis 21 LTS In Java 21 LTS finden sich verschiedene API-Neuerungen. Dabei gibt es mit Sequenced Collections und Virtual Threads zwei Highlights. Sequenced Collections erlauben es, Collections von vorne und hinten zu verarbeiten. Mit Virtual Threads erzielt man eine enorme Vereinfachung beim Multithreading.

Kapitel 11 – JVM-Neuerungen in JDK 18 bis 21 LTS In diesem Kapitel schauen wir unter anderem den Simple Web Server sowie eine Verbesserung bei der Angabe von Code Snippets in JavaDoc an. Abschließend wird das Feature Unnamed Classes and Instance Main Methods behandelt.

Kapitel 12 – Übungen zu den Neuerungen in JDK 18 bis 21 LTS Dieses Kapitel enthält diverse Übungsaufgaben inklusive einer kurzen Darstellung möglicher Lösungen zu den Neuerungen in JDK 18 bis 21 LTS.

Kapitel 13 – Ausblick auf Java 22 Im März 2024 wird Java 22 erscheinen und beinhaltet eine recht stattliche Liste mit 12 JEPs. Spannende Neuerungen sind die State-ments before super(...), die String Templates sowie Unnamed Variables & Pattern und das Foreign Function & Memory API. Auch interessant sind die Stream Gatherers zur Definition eigener Intermediate Operations, die Structured Concurrency zur Verbesserung von Multithreading, die Implicitly Declared Classes and Instance Main Methods zum leichteren Einstieg in Java und abschließend die Scoped Values als Vereinfachung von Wertübergaben.

Kapitel 14 – Zusammenfassung und Schlusswort Dieses Kapitel beginnt mit einigen Gedanken zum Umstieg auf die neuesten Java-Versionen und fasst danach die Themen rund um die vielfältigen Neuerungen bis zum aktuellen Java 21 LTS sowie dem zukünftigen Java 22 noch einmal kurz zusammen.