Michael Inden

Java

Die Neuerungen in Version 17 LTS, 18 und 19

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. Nach rund 1,5 Jahren als freiberuflicher Autor und Trainer ist er seit Januar 2022 als Head of Development in Zürich tätig.

Michael Inden hat über zwanzig Jahre Berufserfahrung beim Entwurf komplexer Softwaresysteme gesammelt und an diversen Fortbildungen sowie 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

Java – die Neuerungen in Version 17 LTS, 18 und 19

dpunkt.verlag

Michael Inden
michael_inden@hotmail.com

Lektorat: Dr. Michael Barabas
Projektkoordinierung: Anja Weimer
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Michael Inden
Herstellung: Stefanie Weidner, Frank Heidt
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-86490-902-3
PDF 978-3-96910-836-9
ePub 978-3-96910-837-6
mobi 978-3-96910-838-3

1. Auflage 2022

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

In Erinnerung an meine geliebte Mutter Marianne Inden
»Wenn die Sonne untergeht, geht sie an anderer Stelle wieder auf.«

Inhaltsverzeichnis

II Ausblick

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 17 LTS (Long Term Support) sowie wesentliche Neuerungen aus den Vorgängern 11 LTS bis 16. Natürlich darf auch ein Blick auf das brandaktuelle Java 18 inklusive eines Ausblicks auf die Neuerungen im zukünftigen Java 19 nicht fehlen.

Zielgruppe

Dies ist kein Buch für Programmierneulinge, sondern richtet sich an diejenigen Leser, die bereits solides Java-Know-how besitzen und sich kurz und prägnant über die wichtigsten Neuerungen in den Java-Versionen 11 bis 17 sowie 18 und 19 informieren wollen. 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 18 oder gar das zukünftige Java 19 in eigenen Projekten erfolgreich gemeistert werden kann. Erleichtert wird ein Umstieg durch eine Vielzahl an Übungen zu den wichtigsten Features, insbesondere denjenigen aus Java 17 LTS.

Um die Beispiele des Buchs möglichst präzise und elegant zu halten, verwende ich diverse Features aus Java 8, 9 und 10. 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 bis 10.

Aufbau dieses Buchs

Nachfolgend möchte ich die Themen der einzelnen Kapitel kurz vorstellen.

Kapitel 1 – Einleitung Die Einleitung stimmt Sie auf Java 11 bis 19 ein und gibt dazu einen Überblick, was Sie so alles in diesem Buch bzw. als Neuerungen erwartet. Zudem thematisiere ich den Programmierstil und gebe Hinweise, welche Voraussetzungen nötig sind, um die aktuellen Java-Versionen mit Build-Tools und IDEs zu verwenden.

Kapitel 2 – Neuerungen in Java 17 im Überblick Dieses Kapitel listet die Erweiterungen aus Java 17 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 3 – Syntaxneuerungen bis Java 17 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 4 – Übungen zu den Syntaxneuerungen in JDK 11 bis 17 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 5 – Neues und Änderungen in den Java-17-APIs In den APIs des JDKs finden sich diverse Neuerungen. Dieses Potpourri umfasst Ergänzungen in der Klasse `String`, kleinere Erweiterungen im Interface `Predicate<T>` sowie der Klasse `Optional<T>` und Convenience-Funktionalitäten in der Utility-Klasse `Files`. Auch das Stream-API bietet Neuerungen unter anderem den sogenannten Teeing-Kollektor sowie die Methode

`mapMulti()`. Als Highlight wird das HTTP/2-API thematisiert.

Kapitel 6 – Übungen zu den API-Erweiterungen in JDK 11 bis 17 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 7 – Änderungen in der JVM bis Java 17 In diesem Kapitel beschäftigen wir uns mit Änderungen in der JVM, etwa bei der Garbage Collection oder der Einführung der `jshell`. Java 11 bringt mit dem Feature »Launch Single-File Source-Code Programs« die Möglichkeit, Java-Klassen ohne explizite vorherige Kompilierung ausführen zu können. Unter anderem sind einige Module zu CORBA, JavaFX usw. nun nicht mehr Bestandteil des JDKs. Darüber hinaus könnte Sie das Microbenchmark-Framework JMH interessieren. Schließlich behandle ich das Tool `jpackage` zum Erzeugen installierbarer Distributionen.

Kapitel 8 – Übungen zu den JVM-Neuerungen in JDK 11 bis 17 Für einige der Neuerungen in der JVM bietet dieses Kapitel ein paar Übungsaufgaben inklusive einer kurzen Darstellung möglicher Lösungen.

Kapitel 9 – Neuerungen in Java 18 Dieses Kapitel bietet einen fundierten Überblick und Einstieg in die Neuerungen aus Java 18, die in jeweils eigenen Abschnitten kurz vorgestellt werden. Dabei werden zunächst die auf JEP (JDK Enhancement Proposal) basierenden Erweiterungen besprochen. Ergänzend werfen wir einen

Blick auf einige Veränderungen in den APIs und schließlich schauen wir uns an, wie wir Java 18 mit Build-Tools und IDEs verarbeiten können.

Kapitel 10 – Ausblick auf Java 19 In diesem Kapitel thematisiere ich die Neuerungen aus Java 19, dessen offizielles Release für den September 2022 angekündigt ist. Derzeit, im Juni 2022, ist der Inhalt (nahezu) final und jeder JEP wird kurz beschrieben und in einigen Fällen mit konkreten Beispielen veranschaulicht. Zudem gehe ich darauf ein, wie Sie das Preview-Release installieren und auf Ihrem Rechner für erste Experimente aktivieren können.

Kapitel 11 – 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 18 sowie dem zukünftigen Java 19 noch einmal kurz zusammen.

Anhang A – Wesentliches aus Java 8, 9 und 10 In Anhang A werden für dieses Buch wesentliche Ergänzungen aus den Java-Versionen 8 bis 10 rekapituliert. Das erleichtert Ihnen das Verständnis der Neuerungen in aktuellen Java-Versionen, selbst dann, wenn Sie sich noch nicht eingehend mit Java 8, 9 oder 10 beschäftigt haben. Neben einer Vorstellung der funktionalen Programmierung mit Lambdas widmen wir uns den Streams, einer wesentlichen Neuerung in JDK 8 zur Verarbeitung von Daten. Abgerundet wird Anhang A durch einen kurzen Blick auf das Date and Time API und verschiedene andere Erweiterungen.

Anhang B – Die Build-Tools Gradle und Maven im Überblick Anhang B liefert eine kurze Einführung in die Build-Tools Gradle und Maven. Ersteres wird auch für die Beispiele dieses Buchs zur Übersetzung genutzt. Mithilfe des vermittelten Wissens sollten Sie dann auch kleinere eigene Projekte mit einem Build-System ausstatten können. Darüber hinaus wird in diesem Anhang das Build-Tool Maven überblicksartig vorgestellt. Es war jahrelang der De-facto-Standard und deswegen lassen sich Maven-Projekte einfacher als andere in die gängigen IDEs importieren.

Sourcecode und ausführbare Programme

Um den Rahmen des Buchs nicht zu sprengen, stellen die Listings häufig nur Ausschnitte aus lauffähigen Programmen dar, wobei wichtige Passagen zum besseren Verständnis mitunter fett hervorgehoben sind. Der vollständige Sourcecode steht auf der Webseite `https://dpunkt.de/produkt/java-die-neuerungen-in-version-17-lts-18-und-19/` zum Download bereit. Neben dem Sourcecode befinden sich auf der Webseite auch mehrere Eclipse-Projekte, über die sich alle Programme ausführen lassen.

Ergänzend wird dort jeweils die Datei `build.gradle` mitgeliefert, die den Ablauf des Builds für Gradle beschreibt. Dieses Build-Tool besitzt viele Vorzüge, wie die kompakte und gut lesbare Notation, und vereinfacht die Verwaltung von Abhängigkeiten enorm. Darüber hinaus erlaubt Gradle das Starten von Programmen, wobei der jeweilige Programmname in Kapitälchenschrift, etwa DateTimeExample, angegeben wird.

Blockkommentare in Listings

Beachten Sie bitte, dass sich in den Listings diverse Blockkommentare 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. Für die Beispiele des 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(final String[] args) throws
InterruptedException,

                        IOException

{

    // Prozess erzeugen

    final String command = "sleep 60s";

    final String commandWin = "cmd timeout 60";

    final Process sleeper =
    Runtime.getRuntime().exec(command);

    ...


    // Process => ProcessHandle

    final ProcessHandle sleeperHandle =
    ProcessHandle.of(sleeper.pid()).

                        orElseThrow(IllegalStateException::n
                        ew);

    ...
```

```
}
```

Konventionen

Verwendete Zeichensätze

In diesem Buch gelten folgende Konventionen bezüglich der Schriftart: Neben der vorliegenden Schriftart werden 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. Namen von Programmen und Entwurfsmustern werden in KAPITÄLCHEN geschrieben. 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.

Tipps und Hinweise aus der Praxis

Dieses Buch ist mit diversen Praxistipps gespickt. In diesen werden interessante Hintergrundinformationen präsentiert oder es wird auf Fallstricke hingewiesen.

Tipp: Praxistipp

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

Verwendete Klassen aus dem JDK

Werden Klassen des JDKs erstmalig im Text erwähnt, so wird deren voll qualifizierter Name, d. h. inklusive der Package-Struktur, angegeben: Die Klasse `String` würde demnach als `java.lang.String` notiert – alle weiteren Nennungen erfolgen dann ohne Angabe des Package-Namens. Diese Regelung erleichtert initial die Orientierung und ein Auffinden im JDK und zudem wird der nachfolgende Text nicht zu sehr aufgebläht. Die voll qualifizierte Angabe hilft insbesondere, da in den Listings eher selten `import`-Anweisungen abgebildet werden.

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 – das gilt ganz besonders für Methoden mit generischen Parametern.

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
(G)UI	(Graphical) User Interface
IDE	Integrated Development Environment
JDK	Java Development Kit
JEP	JDK Enhancement Proposal
JLS	Java Language Specification

JRE	Java Runtime Environment
JSR	Java Specification Request
JVM	Java Virtual Machine

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 zum Gelingen des Buchs beigetragen haben. Insbesondere konnte ich bei der Erstellung des Manuskripts auf ein starkes Team an Korrekturlesern zurückgreifen. Es ist hilfreich, von den unterschiedlichen Sichtweisen und Erfahrungen profitieren zu dürfen.

Zunächst einmal möchte ich mich bei Michael Kulla, der als Trainer für Java SE und Java EE bekannt ist, für sein mehrmaliges, gründliches Review vieler Kapitel und die fundierten Anmerkungen bedanken. Dank Ralph Willenborg enthält der Text doch diverse Tippfehler weniger und hat auch strukturell gewonnen. Sven Friederichs hat an verschiedenen Stellen sprachlich für mehr Stringenz gesorgt und darüber hinaus einige inhaltliche Verbesserungsvorschläge eingebracht. Schließlich gab mir Jonas Hübner noch den einen oder anderen Hinweis. Danke nochmals euch allen!

Ebenso geht ein Dankeschön an das Team des dpunkt.verlags (Dr. Michael Barabas, Anja Weimer und Stefanie Weidner) für die tolle Zusammenarbeit. Außerdem möchte ich mich bei Ursula Zimpfer für ihre Adleraugen beim Copy-Editing bedanken.

Anregungen und Kritik

Trotz großer Sorgfalt und mehrfachen Korrekturlesens lassen sich missverständliche Formulierungen oder 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 Juni 2022
Michael Inden

1 Einleitung

Dieses Buch stellt Ihnen alle wesentlichen Neuerungen aus den LTS-Releases (Long Term Support) Java 11 und 17 vor und behandelt damit natürlich auch die dazwischen liegenden Releases wie Java 12, 13, 14 usw. Darüber hinaus gebe ich einen Ausblick auf das brandaktuelle Java 18 und das zukünftige Java 19.

Alle Leser, die privat oder beruflich noch auf Java 8 setzen, finden in Anhang A einen kompakten Schnelleinstieg zu den herausragendsten bzw. bedeutendsten Änderungen der Java-Versionen 8, 9 sowie 10, um sich für die Erweiterungen ab Java 11 inhaltlich und syntaktisch vorzubereiten.

Bevor wir uns ein wenig mit den in diesem Buch behandelten Themen beschäftigen, möchte ich die Veränderungen der letzten Jahre in der Releasepolitik von Oracle darlegen, weil sich hier einiges getan hat.

1.1 Releasepolitik

Vor Java 9 gab es keinen festen Takt oder Releasezyklus. Stattdessen wurde die Veröffentlichung einer neuen Java-

Version an die Fertigstellung wesentlicher Features geknüpft. Wir alle kennen es noch gut: Früher wurden Java-Releases aufgrund unfertiger Features häufiger verschoben. Um dem entgegenzuwirken, hat Oracle nach dem Erscheinen von Java 9 auf einen halbjährlichen Releasezyklus umgestellt. Das erlaubt es, die jeweils bis zu diesem Zeitpunkt fertig implementierten Funktionalitäten zu veröffentlichen. Allerdings sind logischerweise nicht immer schon alle Features auch so weit, um final veröffentlicht zu werden. Daher gibt es seit Java 9 zwei Besonderheiten: die Preview-Features und die Incubator-Features.

Preview-Features

Sogenannte Preview-Features sind zwar vollständig spezifiziert und implementiert. Sie werden aber zunächst als Vorschau zum Sammeln von Erfahrungen und Feedback ins JDK integriert. Durch die Rückmeldungen möchte man der Funktionalität dann in Folgereleases einen weiteren Feinschliff geben. Allerdings sind Preview-Features standardmäßig nicht zugänglich und müssen explizit über die Angabe von `--enable-preview` sowohl beim Kompilieren als auch beim Ausführen über den gleichnamigen JVM-Aufrufparameter freigeschaltet bzw. aktiviert werden, um die Preview-Features in eigenen Programmen verwenden zu können.

Incubator-Features

Ergänzend zu den Preview-Features gibt es noch sogenannte Incubator-Features, die oftmals in Form korrespondierender Module implementiert und bereitgestellt werden. Auch hierbei geht es um das

Sammeln von Erfahrungen und Feedback, allerdings auf einer vorläufigen Umsetzung. Es kann bei Incubator-Features durchaus sein, dass sich Dinge noch grundlegend ändern oder sogar später vollständig entfernt bzw. gar nicht in ein finales JDK aufgenommen werden. Weil Incubator-Features bzw. die bereitstellenden Module nicht offizieller Bestandteil des JDKs sind, müssen diese speziell aufgenommen werden. Dazu dient der JVM-Aufrufparameter `--add-modules`. Dieser ist interessanterweise nicht beim Kompilieren, sondern nur beim Ausführen anzugeben.

Releasekadenz im Wandel und die Auswirkungen

Kommen wir auf die Releasekadenz zurück. Bis einschließlich Java 9 wurden neue Java-Versionen immer Feature-basiert veröffentlicht. Das hatte in der Vergangenheit oftmals und mitunter auch beträchtliche Verschiebungen des geplanten Releasetermins zur Folge, nämlich immer dann, wenn für die Version wesentliche Features noch nicht fertiggestellt waren. Insbesondere deshalb verzögerten sich die Veröffentlichungen von Java 8 und Java 9 um mehrere Monate bzw. sogar über ein Jahr. Erinnern wir uns: Die Java-Gemeinde musste auf die Veröffentlichung von Java 9 deutlich länger als ursprünglich geplant warten – es gab gleich mehrere Verschiebungen, zunächst von September 2016 auf März 2017, dann auf Juli 2017 und schließlich auf September 2017. Schlimmer noch: Es war damals absolut nicht abzusehen oder vorab zu planen, wann eine neue Java-Version tatsächlich veröffentlicht werden würde.

Mit der Umstellung auf eine zeitbasierte Releasestrategie möchte man derartigen Verzögerungen

entgegenwirken und die Planbarkeit erhöhen, indem jedes halbe Jahr eine neue Java-Version veröffentlicht wird, die all jene Features enthält, die bereits bis zu dem Termin fertig sind. Zudem sollte dann alle drei Jahre eine LTS-Version (Long Term Support) erscheinen. Eine solche ist in etwa vergleichbar mit den früheren Major-Versionen. Das war zwischen Java 11 und 17 dann auch der Fall. Mit Letzterem wurde dann eine Verkürzung des LTS-Zyklus auf 2 Jahre beschlossen, wodurch auch diejenigen Firmen schneller von den Neuerungen profitieren, die lediglich von LTS zu LTS springen wollen oder können.

Lassen Sie mich noch auf Folgendes hinweisen: Zwar kann die schnelle, halbjährliche Releasefolge eine größere Herausforderung für Toolhersteller sein, für uns als Entwickler ist es aber oftmals positiv, weil wir potenziell weniger lang auf neue Features warten müssen. Das konnte früher recht zermürbend sein.

Allerdings gilt das Positive vor allem für eigene Hobbyprojekte, weil man dort mit den Neuerungen experimentieren kann und weniger durch Restriktionen eingeschränkt ist. Im professionellen Einsatz wird man eher auf Kontinuität und die Verfügbarkeit von Security Updates setzen, weshalb in diesem Kontext vermutlich nur LTS-Versionen in Betracht kommen, um Migrationsaufwände kalkulierbar und besser planbar zu halten.

Tipp: Am Rande …