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. Auf der Webseite zu diesem Buch `www.dpunkt.de/java-9-14-die-neuerungen` steht dann der vollständige, kompilierbare Sourcecode zu den Programmen zum Download bereit. Neben dem Sourcecode befindet sich auf der Webseite auch ein Eclipse-Projekt, über das sich alle Programme ausführen lassen.

Ergänzend wird 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::new);
    ...
}
```

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. Ebenfalls bin ich Prof. Dr. Dominik Gruntz sehr dankbar für diverse Verbesserungsvorschläge in den Kapiteln zu Java 12 und 14. Zudem erhielt ich den einen oder anderen Tipp von Jean-Claude Brantschen sowie noch in letzter Minute von Prof. Dr. Carsten Kern.

Nachfolgende Danksagung bezieht sich auf den Java-9-Teil sowie die Texte zur Modularisierung. Merten Driemeyer, Dr. Clemens Gugenberger, Prof. Dr. Carsten Kern sowie Andreas Schöneck haben mit verschiedenen hilfreichen Anmerkungen zu einer Verbesserung beigetragen. Zudem hat Ralph Willenborg dort mal wieder ganz genau gelesen und so diverse Tippfehler gefunden. Vielen Dank dafür! Auch von Albrecht Ermgassen erhielt ich den einen oder anderen Hinweis.

Schließlich bedanke ich mich bei einigen ehemaligen Arbeitskollegen der Zühlke Engineering AG: Jeton Memeti und Marius Reusch trugen durch ihre Kommentare zur

Klarheit und Präzisierung bei. Auch von Hermann Schnyder von der Swisscom erhielt ich ein paar Anregungen.

Ebenso geht ein Dankeschön an das Team des dpunkt.verlags (Dr. Michael Barabas, Martin Wohlrab, Anja Weimer und Stefanie Weidner) 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. Glücklicherweise musste sie beim Entstehen dieses Buchs zu den Neuerungen in Java 9 bis 14 einen weit weniger gestressten Autor ertragen, als dies früher beim Schreiben meines Buchs »Der Weg zum Java-Profi« der Fall war.

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 März 2020
Michael Inden

1 Einleitung

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. Zwar kann diese schnelle 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 mühsam sein, wie die letzten Jahre gezeigt haben. Ein paar Gedanken dazu greift der nachfolgende Hinweiskasten auf.

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 zeitlich besser planbar zu halten.

Hinweis: Oracles neue Releasepolitik

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, wenn für die Version wesentliche Features noch nicht fertig waren. Insbesondere deshalb verzögerten sich Java 8 und Java 9 um mehrere Monate bzw. sogar über ein Jahr: Rund 3,5 Jahre nach dem Erscheinen von JDK 8 im März 2014 ging Java mit Version 9 im September 2017 an den Start. Wieder einmal musste die Java-Gemeinde auf die Veröffentlichung der Version 9 des JDKs länger warten – es gab gleich mehrere Verschiebungen, zunächst von September 2016 auf März 2017, dann auf Juli 2017 und schließlich auf September 2017.

Mit einer zeitbasierten Releasestrategie möchte man derartigen Verzögerungen entgegenwirken, indem jedes halbe Jahr eine neue Java-Version veröffentlicht wird, die all jene Features enthält, die bereits fertig sind. Alle drei Jahre ist dann eine LTS-Version (Long Term Support) geplant. Eine solche ist in etwa vergleichbar mit den früheren Major-Versionen.