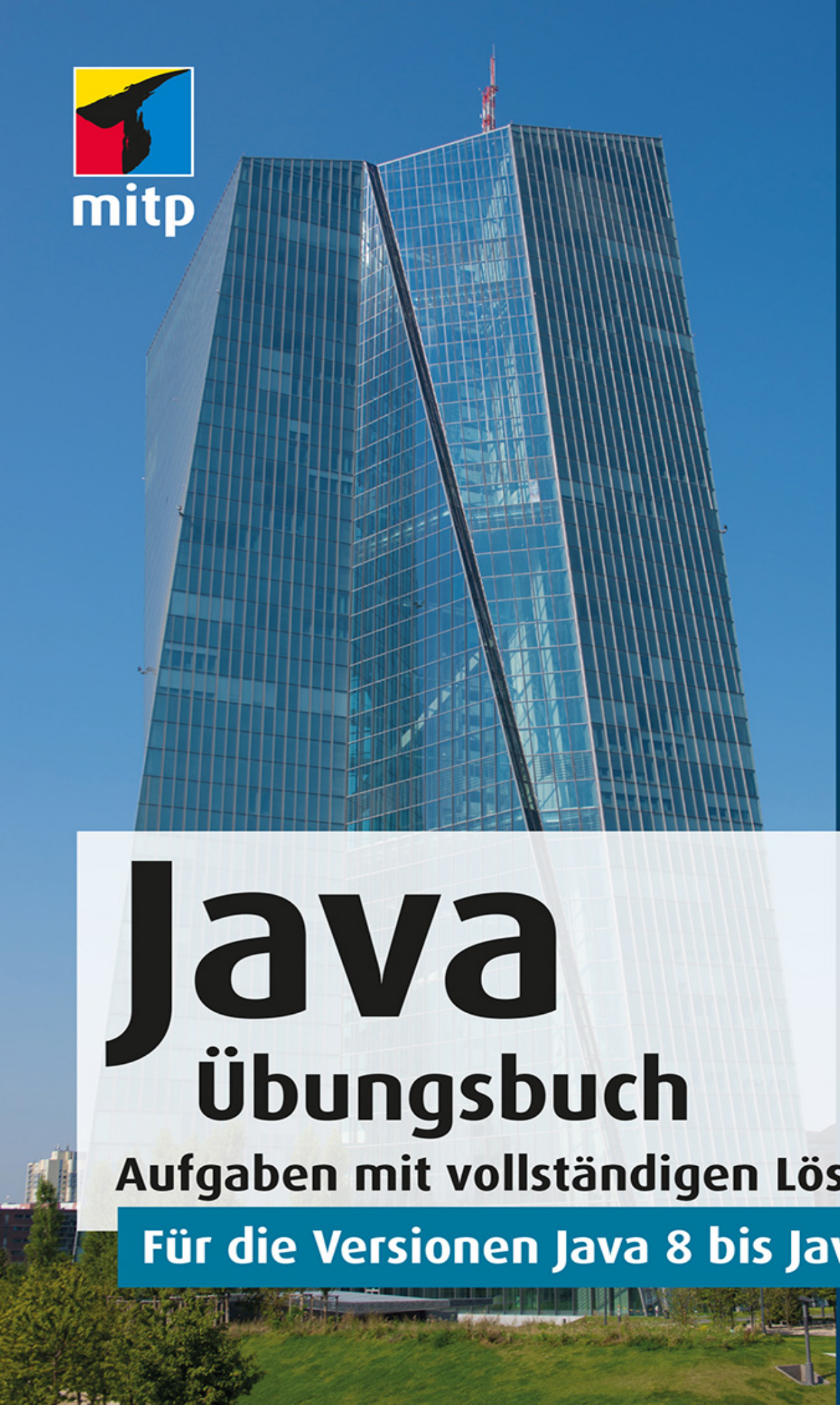

mitp

Elisabeth Jung

Java Übungsbuch

Aufgaben mit vollständigen Lösungen

Für die Versionen Java 8 bis Java 17

Inhaltsverzeichnis

Elisabeth Jung

Java Übungsbuch

Für die Versionen Java 8 bis Java 17

Aufgaben mit vollständigen Lösungen

Impressum

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 978-3-7475-0451-2
1. Auflage 2021

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Sabine Schulz
Sprachkorrektorat: Petra Heubach-Erdmann
Covergestaltung: © maxec / stock.adobe.com
electronic **pub**lication: III-satz, Husby, www.drei-satz.de

Einleitung

Das Java Übungsbuch: Für die Versionen Java 8 bis Java 17 ist wie alle meine Übungsbücher aus der Erkenntnis entstanden, dass zu umfangreiche Beispiele mit komplizierten Algorithmen beim Lernen von Java am Anfang keine echte Hilfe bieten. Darum liegt der Schwerpunkt des Buches nicht auf der Umsetzung von komplizierten Vorgängen, sondern konzentriert sich stattdessen darauf, die in der Dokumentation nicht immer verständlich formulierten Erläuterungen zu Java-Klassen und -Interfaces mit einfachen Beispielen zu erklären und gleichzeitig die zugrunde liegenden Konzepte zu erörtern.

Das Java Übungsbuch: Für die Versionen Java 8 bis Java 17 wendet sich in erster Linie an Lehrer, Schüler und Studenten als Begleitliteratur zum Lernen der Programmiersprache Java, ist aber auch zum Selbststudium für alle Interessenten an dem Erlernen der Programmiersprache geeignet.

Durch die Einfachheit und Vollständigkeit der Aufgabenlösungen sowie die unterschiedlichen Lösungsmöglichkeiten erhält der Leser ein fundiertes Verständnis für die Aufgabenstellungen und deren Lösungen.

Durch das Lösen von Aufgaben soll der in Referenz- und Lehrbüchern von Java angebotene Stoff vertieft werden, und die dabei erzielten Ergebnisse können anhand der Lösungsvorschläge überprüft werden. Die Beispiele im Buch sind eher selten von zu komplexer Natur, sodass der eigentliche Zweck nicht in den Hintergrund tritt, und alle beschriebenen Themen können tiefgehend und präzise damit eingeübt werden.

Vorkenntnisse

Es ist Voraussetzung, dass der Leser zusätzlich mit einem Lehrbuch zu Java arbeitet bzw. bereits damit gearbeitet hat. Die grundlegenden Erläuterungen zu Java in diesem Buch können lediglich als Wiederholung des bereits vorhandenen Wissens dienen, reichen aber nicht aus, um die Sprache Java erst neu zu lernen.

Als weitere Voraussetzung gelten Grundlagen im Bereich der Programmierung und im Umgang mit dem Betriebssystem. Ein paralleler Zugriff auf die Java-Online-Dokumentation kann Hilfe zu den Java-Standard-Klassen bieten.

Aufbau des Buches

Jedes Kapitel beginnt mit einer kurzen und knappen Wiederholung des Stoffes, der in den Übungsaufgaben dieses Kapitels verwendet wird. Danach folgen alle Aufgabenstellungen der Übungen. Am Ende des Kapitels stehen gesammelt die Lösungen der Übungsaufgaben mit Kommentaren, Erläuterungen und Hinweisen.

Die Aufgaben haben unterschiedliche Schwierigkeitsgrade. Dieser wird im Aufgabenkopf durch ein bis drei Sternchen gekennzeichnet:

ein Sternchen für besonders einfache Aufgaben, die auch von Anfängern leicht bewältigt werden können

zwei Sternchen für etwas kompliziertere Aufgaben, die einen durchschnittlichen Aufwand benötigen

drei Sternchen für Aufgaben, die sich an geübte Programmierer richten und einen wesentlich höheren Aufwand oder die Kenntnis von speziellen Details erfordern

Die Programme aus früheren Übungen werden teilweise in späteren Übungen gebraucht und es wird auch immer wieder auf theoretische Zusammenhänge zurückgekommen oder hingewiesen.

Die Lösungsvorschläge haben umfangreiche Kommentare, sodass ein Verständnis für die durchgeführte Aufgabe auch daraus abgeleitet werden kann und dadurch jede einzelne Aufgabe im Gesamtkontext unabhängig erscheint.

In den Kapiteln 1, 2 und 3 liegt das Hauptmerkmal auf den Eigenheiten der objektorientierten Programmierung mit Java. Durch eine Vielzahl von Beispielen wird gezeigt, was die Java-Standard-Klassen und Interfaces an Funktionalitäten bieten und wie diese sinnvoll in die Definition von eigenen Klassen eingebettet werden können. Diese Kapitel enthalten zusätzlich Informationen zur Reflection-API von Java, der Definition von Annotationen und inneren Klassen sowie Neuerungen aus den Versionen 8 bis 13, die sich auf die neue Date&Time-API, Textblöcke, Compact Strings und die Weiterentwicklung von Interfaces beziehen. Mit Java 8 wurden sogenannte Default-Methoden eingeführt. Diese werden in der Literatur auch als »virtual extension«- bzw. »defender«-Methoden bezeichnet und

Schnittstellen, die über derartige Methoden verfügen, als erweiterte Schnittstellen. Damit können Interfaces zusätzlich zu abstrakten Methoden konkrete Methoden in Form von Standard-Implementierungen definieren und in Java wird die Mehrfachvererbung von Funktionalität ermöglicht. Neben Default-Methoden können Interfaces in Java nun auch statische Methoden enthalten. Anders als die statischen Methoden von Klassen werden diese jedoch nicht von abgeleiteten Typen geerbt.

Kapitel 4 beschäftigt sich im Detail mit Generics und dem Collection Framework mit all seinen generischen Klassen und Interfaces sowie mit der Definition von Enumerationen. Die Typinferenz für Methoden und beim Erzeugen von generischen Typen (der Diamond-Operator) sowie das Subtyping von parametrisierten und Wildcard-parametrisierten Typen sind ebenfalls Gegenstand der Themen aus diesem Kapitel.

Kapitel 5 erläutert das Exception-Handling.

Kapitel 6 beschäftigt sich mit den neuen Sprachmitteln von Java 8, Lambdas und Streams sowie mit weiteren Neuerungen aus den Versionen 8 bis 14, wie Switch-Expressions und Local Variable Type Inference.

Mit der Java-Version 8 haben sich ganz neue Betrachtungsweisen und Programmiertechniken in der Entwicklung von Applikationen mit Java eröffnet. Eine der wichtigsten Neuerungen in Java 8 sind neue Sprachmittel, die sogenannten Lambda-Ausdrücke, eine Art anonyme Methoden, die auf funktionalen Interfaces basieren. Diese besitzen jedoch eine viel kompaktere Syntax als Methoden. Das resultiert daraus, dass in ihrer Benutzung auf Namen, Modifikatoren, Rückgabetyp, `throws`-Klausel und in vielen Fällen auch auf Parameter verzichtet werden kann. Mit ihnen

kann Funktionalität ausgeführt, gespeichert und übergeben werden, wie dies bisher nur von Instanzen in Java bekannt war.

Damit verbundene Themen wie die Gegenüberstellung zu anonymen Klassen, Syntax und Semantik, Behandlung von Exceptions, Scoping und Variable Capture, Methoden- und Konstruktor-Referenzen werden in den ersten Unterkapiteln des 6. Kapitels dieses Buches beschrieben und anhand von vielen Beispielen erläutert.

Des Weiteren finden Sie hier die Beschreibung aller neuen funktionalen Interfaces und deren Methoden. Die nachfolgenden Unterkapitel beschäftigen sich im Detail mit der Definition und Nutzung von Streams. Ein Stream besteht aus einer Folge von Werten (in der Literatur wird auch von Sequenzen von Elementen gesprochen), die nur teilweise von mehreren in einer Pipeline dazwischenliegenden Operationen ausgewertet und durch eine abschließende Operation bereitgestellt werden. Diese Operationen werden in Java als Methodenaufrufe formuliert, die Funktionalität in Form von Lambdas und Methoden-Referenzen entgegennehmen können und diese auf alle Elemente der Folge anwenden.

Mit einer Vielzahl von Aufgaben basierend auf Lambdas, Streams und Kollektoren (in denen Stream-Elemente angesammelt und reduziert werden können) werden die neuen Techniken angewandt und alle neuen Begriffe erklärt.

Kapitel 7 präsentiert das neue Java-Modulsystem. Mit dem neuen Modulsystem wurde Java selbst modular gemacht und es können eigene Applikationen und Bibliotheken modularisiert werden.

Java 9 führt das Modul als eine neue Programmkomponente ein. Das Erzeugen von Modulen und deren Abhängigkeiten führen dazu, dass der Zugriffsschutz in Java 9 restriktiver ist. Das Anlegen der erforderlichen Verzeichnisstrukturen für modulbasierte Applikationen, das Packaging von Modul-Code sowie die Implementierung von Services werden ebenfalls im Detail erklärt. Eine Vielzahl von Applikationen mit ausführlichen `.cmd`-Dateien für deren Ausführung ergänzen die theoretischen Erläuterungen aus diesem Kapitel.

In Kapitel 8 werden die Weiterentwicklungen aus der Programmiersprache mit den Versionen 14 bis 17 erläutert. Dazu gehören die Einführung von Records und Sealed Classes sowie das Pattern Matching.

Records wurden in der Version 14 entworfen, um die Wiederholungen von repetitivem Code in Datenklassen zu unterdrücken. Sie überlassen dem Compiler eine korrekte Generierung der Methoden `equals()`, `hashCode()`, `toString()` (die in Klassen, um eine Wertegleichheit von Objekten zu ermöglichen, überschrieben werden müssen) und von Zugriffsmethoden.

`record`-Klassen helfen in Kombination mit den in Java 15 neu eingeführten `sealed`-Klassen und -Interfaces, die auch mit Java 16 im Preview-Status bleiben und mit Java 17 finalisiert werden, die funktionalen Features von Java zu erweitern, insbesondere das Pattern Matching und in naher Zukunft die Destrukturierung von Objekten.

Sealed Classes und Interfaces sind Java-Datentypen, für die die Definition von Subtypen reduziert wird. Sie können nur von den in ihrer Deklaration angegebenen Typen erweitert bzw. implementiert werden.

Auch wenn es keine direkten Abhängigkeiten zwischen den Previews aus den JEPs 395, 394, 409, 406 und 405 gibt, die die Einführung dieser neuen Java-Datentypen sowie das Pattern Matching in Java beschreiben, so sind die von diesen vorgeschlagenen neuen Java-Features, wie mit vielen Beispielen in den Kapiteln 8 und 9 illustriert wird, sehr gut zusammen einsetzbar und im weitesten Sinne auch dafür gedacht.

Das Pattern Matching wurde in Java ursprünglich für den Abgleich von regulären Ausdrücken mit einem Text eingesetzt und für einen Vergleich von Typen im Zusammenhang mit dem `instanceof`-Operator und `switch` weiterentwickelt.

Der `instanceof`-Operator wurde erweitert, sodass anstelle eines Typ-Tests ein Musterabgleich-Test (»type test pattern«) durchgeführt wird. Dieser prüft die Übereinstimmung eines Zielobjekts mit einem vorgegebenen Mustertyp und erweist sich als sehr nützlich beim Schreiben von `equals()`-Methoden.

Mit Java 17 sind rund um das Pattern Matching weitere Funktionen im Zusammenhang mit Switch Statements und Switch Expressions realisiert worden. Damit werden die Restriktionen für den Typ des Ausdrucks, der im `switch` übergeben wird, weitestgehend aufgehoben. Bei einem klassischen `switch` waren zugelassen: ganzzahlige primitive Typen (`char`, `byte`, `short`, `int`) und die dazugehörigen Wrapper-Typen (`Character`, `Byte`, `Short`, `Integer`) sowie `String` und `enum`-Konstanten. Diese Auswahl wurde nun auf ganzzahlige primitive Typen und beliebige Referenztypen erweitert, sodass `class`-, `enum`-, `record`- und `array`-Typen zugelassen sind, die zusammen mit einem `null-case`-Label und einem `default`-Label die Angaben in den `switch-case`-Labels ausmachen können.

Die Destrukturierung von Objekten wird zusammen mit Record Patterns und Array Patterns (JEP 405) die Entwickler von nachfolgenden Java-Versionen weiter beschäftigen.

Neu in dieser Auflage des Buches sind Tests mit JUnit 5 und Gradle, die in Kapitel 9 beispielhaft präsentiert werden.

JUnit 5 kann von der Website `https://junit.org/junit5/` unter »Latest Release« (aktuelle Version zum Zeitpunkt der Redaktion dieses Buches waren: Jupiter v5.7.1, Vintage v5.7.1, Platform v1.7.1) heruntergeladen werden.

Zum Testen von Applikationen werden, wie auch in den bevorstehenden Versionen von JUnit üblich, sogenannte Testklassen geschrieben. Sie beinhalten Methoden, die Testfälle beschreiben, den Rückgabetyp `void` aufweisen und durch Annotationen gekennzeichnet sind.

JUnit 5 führt darüber hinaus das Konzept eines ConsoleLaunchers ein, der benutzt werden kann, um Tests zu entwickeln, zu filtern und durchzuführen.

Um Ihnen ein gutes Verständnis für Details zu ermöglichen, wähle ich in diesem Buch die Ausführung über die Kommandozeile, die der ConsoleLauncher in diesem Fall ermöglicht.

Sicherlich sind Build-Tools wie Gradle und IDEs wie Eclipse, Intellij IDEA oder Maven eine große Hilfe nicht nur bei der Ausführung von JUnit-Tests, sondern generell in der Programmierung mit Java. Die Angabe von Details in diesem Zusammenhang würde den Rahmen dieses Buches jedoch sprengen.

In einem Unterkapitel in Kapitel 9 erfolgt eine kurze Beschreibung von Gradle und der Ausführung von Tests mit diesem Tool. Weil es gerade im Zusammenhang mit JUnit-

Tests dem Anwender viel Kopfzerbrechen und Arbeit erspart, präsentiere ich es als Alternative zum ConsoleLauncher für die Durchführung von JUnit-Tests für die Java-Applikationen.

Eine neue Gradle-Version kann von der Website `https://gradle.org/releases/` heruntergeladen werden. Zum Zeitpunkt der Buchredaktion war die Version `v7.0.1` aktuell.

Weil der Schwerpunkt des Buches nicht auf der Umsetzung von aufwendigen Algorithmen liegen soll, verwende ich einfache Beispiele mit Zahlen, Buchstaben, Wörtern, Büchern, Wochentagen, geometrischen Figuren etc. und teilweise auch mit ganz abstrakten Klassennamen wie `Klasse1`, `Klasse2`, `KlasseA`, `KlasseB` etc.

An dieser Stelle möchte ich auf das dem Buch zugrunde liegende Konzept hinweisen, dass parallel zu einfachen Aufgaben, die zu allen eingeführten Definitionen und Begriffen gebracht werden, auch Aufgaben von einem höheren Schwierigkeitsgrad präsentiert werden. Dabei werden anhand von inhaltlichen Zusammenhängen zwischen den Beispielen viele Basiskonzepte von Java erläutert.

Ich habe generell versucht, keine Begriffe, Klassen und Komponenten zu benutzen, die nicht schon in vorangehenden Beispielen und Kapiteln definiert oder erläutert wurden. In den wenigen Fällen, wo es sich nicht vermieden ließ, wird darauf hingewiesen und auf die entsprechenden Stellen verwiesen.

Das Buch soll möglichst parallel zu einer Vielzahl von Java-Lehrbüchern eingesetzt werden können und einen Beitrag dazu leisten, die große Fülle von Informationen, die auf uns über die API-Dokumentation zukommt, besser einzuordnen und korrekt anwenden zu können.