



More Java 17

An In-Depth Exploration of the
Java Language and Its Features

—
Third Edition
—

Kishori Sharan
Peter Späth

Apress®

More Java 17

**An In-Depth Exploration of the Java
Language and Its Features**

Third Edition

Kishori Sharan

Peter Späth

Apress®

More Java 17: An In-Depth Exploration of the Java Language and Its Features

Kishori Sharan
Montgomery, AL, USA

Peter Späth
Leipzig, Sachsen, Germany

ISBN-13 (pbk): 978-1-4842-7134-6
<https://doi.org/10.1007/978-1-4842-7135-3>

ISBN-13 (electronic): 978-1-4842-7135-3

Copyright © 2021 by Kishori Sharan, Peter Späth

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: Matthew Moodie
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image by Ben Kolde on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484271346. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

To Paulina

Table of Contents

About the Authors	xix
About the Technical Reviewers	xxi
Introduction	xxiii
Chapter 1: Annotations	1
What Are Annotations?.....	1
Declaring an Annotation Type.....	6
Restrictions on Annotation Types	10
Restriction #1	10
Restriction #2	11
Restriction #3	12
Restriction #4	12
Restriction #5	13
Restriction #6	13
Default Value of an Annotation Element.....	14
Annotation Type and Its Instances	15
Using Annotations	17
Primitive Types	17
String Types	18
Class Types	19
Enum Type	21
Annotation Type	23
Array Type Annotation Element.....	24
No Null Value in an Annotation	25
Shorthand Annotation Syntax.....	25
Marker Annotation Types.....	28

TABLE OF CONTENTS

- Meta-Annotation Types 29
 - The Target Annotation Type..... 29
 - The Retention Annotation Type 33
 - The Inherited Annotation Type 35
 - The Documented Annotation Type 36
 - The Repeatable Annotation Type 37
- Commonly Used Standard Annotations..... 38
 - Deprecating APIs 39
 - Suppressing Named Compile-Time Warnings 54
 - Overriding Methods 55
 - Declaring Functional Interfaces..... 56
- Annotating Packages 58
- Annotating Modules 58
- Accessing Annotations at Runtime 59
- Evolving Annotation Types..... 66
- Annotation Processing at Source Code Level 66
- Summary..... 76
- Exercises..... 77
- Chapter 2: Reflection 81**
 - What Is Reflection? 81
 - Reflection in Java..... 82
 - Loading a Class..... 84
 - Using Class Literals 85
 - Using the `Object::getClass()` Method 86
 - Using the `Class::forName()` Method..... 86
 - Class Loaders..... 90
 - Reflecting on Classes..... 93
 - Reflecting on Fields 100
 - Reflecting on Executables..... 103
 - Reflecting on Methods..... 106
 - Reflecting on Constructors 109

Creating Objects.....	111
Invoking Methods.....	113
Accessing Fields	114
Deep Reflection.....	117
Deep Reflection Within a Module.....	119
Deep Reflection Across Modules	124
Deep Reflection and Unnamed Modules.....	132
Deep Reflection on JDK Modules	132
Reflecting on Arrays.....	135
Expanding an Array	138
Who Should Use Reflection?	139
Summary.....	140
Exercises.....	141
Chapter 3: Generics	145
What Are Generics?.....	145
Supertype-Subtype Relationship	151
Raw Types	152
Unbounded Wildcards	153
Upper-Bounded Wildcards	157
Lower-Bounded Wildcards	159
Generic Methods and Constructors.....	163
Type Inference in Generic Object Creation	165
No Generic Exception Classes.....	170
No Generic Anonymous Classes.....	170
Generics and Arrays	170
Runtime Class Type of Generic Objects.....	172
Heap Pollution.....	173
Varargs Methods and Heap Pollution Warnings	174
Summary.....	176
Exercises.....	177

Chapter 4: Lambda Expressions 183

- What Is a Lambda Expression? 183
- Why Do We Need Lambda Expressions? 186
- Syntax for Lambda Expressions..... 188
 - Omitting Parameter Types 190
 - Using Local Variable Syntax for Parameters..... 191
 - Declaring a Single Parameter..... 191
 - Declaring No Parameters..... 192
 - Parameters with Modifiers 192
 - Declaring the Body of Lambda Expressions 192
- Target Typing..... 193
- Functional Interfaces 204
 - Using the `@FunctionalInterface` Annotation..... 205
 - Generic Functional Interface 206
 - Intersection Type and Lambda Expressions 208
 - Commonly Used Functional Interfaces 209
 - Using the `Function<T,R>` Interface 211
 - Using the `Predicate<T>` Interface..... 213
 - Using Functional Interfaces 215
- Method References 221
 - Static Method References 224
 - Instance Method References..... 227
 - Supertype Instance Method References..... 231
 - Constructor References..... 234
 - Generic Method References 237
- Lexical Scoping..... 239
- Variable Capture..... 242
- Jumps and Exits..... 246
- Recursive Lambda Expressions 247
- Comparing Objects..... 249
- Summary..... 251
- Exercises..... 252

Chapter 5: Threads	257
What Is a Thread?	257
Creating Threads in Java	262
Specifying Your Code for a Thread	265
Inheriting Your Class from the Thread Class	265
Implementing the Runnable Interface.....	266
Using a Method Reference	267
A Quick Example.....	267
Using Multiple Threads in a Program	268
Issues in Using Multiple Threads	270
Java Memory Model.....	274
Atomicity	276
Visibility	276
Ordering.....	277
Object's Monitor and Thread Synchronization	277
Rule #1	288
Rule #2	288
The Producer/Consumer Synchronization Problem	296
Which Thread Is Executing?.....	301
Letting a Thread Sleep	302
I Will Join You in Heaven	304
Be Considerate to Others and Yield.....	307
Lifecycle of a Thread	308
Priority of a Thread.....	313
Is It a Demon or a Daemon?.....	315
Am I Interrupted?.....	318
Threads Work in a Group.....	323
Volatile Variables.....	324
Stopping, Suspending, and Resuming Threads.....	327
Spin-Wait Hints	333

TABLE OF CONTENTS

- Handling an Uncaught Exception in a Thread 335
- Thread Concurrency Packages 337
- Atomic Variables 337
 - CAS 338
 - Scalar Atomic Variable Classes 339
 - Atomic Array Classes 339
 - Atomic Field Updater Classes 340
 - Atomic Compound Variable Classes 340
- Explicit Locks 342
- Synchronizers 350
 - Semaphores 350
 - Barriers 355
 - Phasers 360
 - Latches 372
 - Exchangers 375
- The Executor Framework 381
 - Result-Bearing Tasks 389
 - Scheduling a Task 392
 - Handling Uncaught Exceptions in a Task Execution 396
 - Executor's Completion Service 399
- The Fork/Join Framework 403
 - Steps in Using the Fork/Join Framework 405
 - A Fork/Join Example 406
- Thread-Local Variables 410
- Setting Stack Size of a Thread 414
- Summary 414
- Exercises 416

Chapter 6: Streams	419
What Are Streams?	420
Streams Have No Storage	421
Infinite Streams	421
Internal Iteration vs. External Iteration	421
Imperative vs. Functional	423
Stream Operations.....	424
Ordered Streams	426
Streams Are Not Reusable.....	427
Architecture of the Streams API	427
A Quick Example	429
Creating Streams	435
Streams from Values	435
Empty Streams	439
Streams from Functions.....	439
Streams from Arrays.....	447
Streams from Collections	447
Streams from Files	448
Streams from Other Sources	450
Representing an Optional Value	451
Applying Operations to Streams	459
Debugging a Stream Pipeline	460
Applying the ForEach Operation	462
Applying the Map Operation	464
Flattening Streams	466
Applying the Filter Operation	469
Applying the Reduce Operation	473
Collecting Data Using Collectors	485
Collecting Summary Statistics	491
Collecting Data in Maps	494
Joining Strings Using Collectors	497

TABLE OF CONTENTS

- Grouping Data 499
- Partitioning Data 504
- Adapting the Collector Results..... 506
- Finding and Matching in Streams 511
- Parallel Streams..... 514
- Summary..... 517
- Exercises..... 518

- Chapter 7: Implementing Services 523**
 - What Is a Service? 523
 - Discovering Services 526
 - Providing Service Implementations 528
 - Defining the Service Interface 530
 - Obtaining Service Provider Instances 531
 - Defining the Service..... 535
 - Defining Service Providers..... 539
 - Defining a Default Prime Service Provider 539
 - Defining a Faster Prime Service Provider..... 541
 - Defining a Probable Prime Service Provider..... 544
 - Testing the Prime Service 546
 - Testing Prime Service in Legacy Mode 552
 - Summary..... 555
 - Exercises..... 556

- Chapter 8: Network Programming..... 559**
 - What Is Network Programming? 560
 - Network Protocol Suite 562
 - IP Addressing Scheme 566
 - IPv4 Addressing Scheme 568
 - IPv6 Addressing Scheme 571
 - Special IP Addresses..... 573
 - Loopback IP Address 574

Unicast IP Address.....	576
Multicast IP Address.....	576
Anycast IP Address.....	577
Broadcast IP Address.....	577
Unspecified IP Address.....	578
Port Numbers.....	578
Socket API and Client-Server Paradigm.....	580
The Socket Primitive.....	583
The Bind Primitive.....	584
The Listen Primitive.....	584
The Accept Primitive.....	584
The Connect Primitive.....	584
The Send/Sendto Primitive.....	586
The Receive/ReceiveFrom Primitive.....	586
The Close Primitive.....	586
Representing a Machine Address.....	587
Representing a Socket Address.....	590
Creating a TCP Server Socket.....	592
Creating a TCP Client Socket.....	598
Putting a TCP Server and Clients Together.....	601
Working with UDP Sockets.....	602
Creating a UDP Echo Server.....	607
A Connected UDP Socket.....	613
UDP Multicast Sockets.....	614
URI, URL, and URN.....	618
URI and URL As Java Objects.....	625
Accessing the Contents of a URL.....	630
Non-blocking Socket Programming.....	641
Socket Security Permissions.....	658
Asynchronous Socket Channels.....	660
Setting Up an Asynchronous Server Socket Channel.....	662

TABLE OF CONTENTS

- Setting Up an Asynchronous Client Socket Channel..... 672
- Putting the Server and the Client Together..... 678
- Datagram-Oriented Socket Channels..... 679
 - Creating the Datagram Channel 679
 - Setting the Channel Options..... 680
 - Sending Datagrams..... 682
- Multicasting Using Datagram Channels..... 686
 - Creating the Datagram Channel 686
 - Setting the Channel Options..... 686
 - Binding the Channel 686
 - Setting the Multicast Network Interface..... 687
 - Joining the Multicast Group 689
 - Receiving a Message..... 689
 - Closing the Channel..... 690
- Further Reading 695
- Summary..... 695
- Exercises..... 697
- Chapter 9: Java Remote Method Invocation..... 699**
 - What Is Java Remote Method Invocation? 700
 - The RMI Architecture..... 702
 - Developing an RMI Application 704
 - Writing the Remote Interface 705
 - Implementing the Remote Interface..... 706
 - Writing the RMI Server Program..... 709
 - Writing the RMI Client Program 713
 - Separating the Server and Client Code..... 716
 - Running the RMI Application..... 717
 - Running the RMI Registry..... 718
 - Running the RMI Server..... 719
 - Running an RMI Client Program 721
 - Troubleshooting an RMI Application 721

java.rmi.server.ExportException	722
java.security.AccessControlException	723
java.lang.ClassNotFoundException.....	723
Debugging an RMI Application	725
Dynamic Class Downloading.....	727
Garbage Collection of Remote Objects.....	730
Summary.....	735
Exercises.....	736
Chapter 10: Scripting in Java	739
What Is Scripting in Java?	740
Installing Script Engines in Maven.....	741
Executing Your First Script.....	743
Using Other Scripting Languages	746
Exploring the javax.script Package	749
The ScriptEngine and ScriptEngineFactory Interfaces	749
The AbstractScriptEngine Class.....	750
The ScriptEngineManager Class.....	750
The Compilable Interface and the CompiledScript Class.....	750
The Invocable Interface	750
The Bindings Interface and the SimpleBindings Class	750
The ScriptContext Interface and the SimpleScriptContext Class.....	751
The ScriptException Class	751
Discovering and Instantiating Script Engines.....	751
Executing Scripts	753
Passing Parameters	755
Passing Parameters from Java Code to Scripts	755
Passing Parameters from Scripts to Java Code	758
Advanced Parameter Passing Techniques	760
Bindings.....	760
Scope.....	762

TABLE OF CONTENTS

- Defining the Script Context..... 763
- Putting Them Together..... 769
- Using a Custom ScriptContext 777
- Return Value of the eval() Method..... 781
- Reserved Keys for Engine Scope Bindings 783
- Changing the Default ScriptContext..... 784
- Sending Script Output to a File 785
- Invoking Procedures in Scripts 787
- Implementing Java Interfaces in Scripts 792
- Using Compiled Scripts..... 798
- Using Java in Scripting Languages..... 801
 - Declaring Variables..... 801
 - Importing Java Classes 802
- Implementing a Script Engine..... 802
 - The Expression Class..... 804
 - The JKScriptEngine Class..... 811
 - The JKScriptEngineFactory Class..... 813
 - Packaging the JKScript Files..... 815
 - Using the JKScript Script Engine..... 815
- JavaFX in Groovy..... 819
- Summary..... 823
- Exercises..... 824
- Chapter 11: Process API 825**
 - What Is the Process API? 825
 - Knowing the Runtime Environment..... 827
 - The Current Process..... 830
 - Querying the Process State..... 830
 - Comparing Processes 835
 - Creating a Process 836
 - Obtaining a Process Handle 855

Terminating Processes.....	858
Managing Process Permissions.....	859
Summary.....	863
Exercises.....	864
Chapter 12: Packaging Modules.....	867
The JAR Format.....	868
What Is a Multi-release JAR?.....	868
Creating Multi-release JARs.....	871
Rules for Multi-release JARs.....	879
Multi-release JARs and JAR URL.....	881
Multi-release Manifest Attribute.....	881
The JMOD Format.....	882
Using the jmod Tool.....	882
Summary.....	891
Exercises.....	892
Chapter 13: Custom Runtime Images.....	893
What Is a Custom Runtime Image?.....	893
Creating Custom Runtime Images.....	894
Binding Services.....	900
Using Plugins with the jlink Tool.....	904
The jimage Tool.....	905
Summary.....	908
Exercises.....	908
Chapter 14: Miscellanea.....	911
Deleted Chapters from Previous Editions.....	911
More JDK17 Novelties.....	912
Local Variables with Automatic Types.....	912
Launch Single-File Source Code Programs.....	914
Enhanced switch Statements.....	914
Text Blocks.....	916

TABLE OF CONTENTS

Enhanced instance of Operator 917

Value Classes: Records..... 918

Sealed Classes 922

Summary..... 922

Appendix: Solutions to the Exercises 923

Exercises in Chapter 1 923

Exercises in Chapter 2 924

Exercises in Chapter 3 925

Exercises in Chapter 4 925

Exercises in Chapter 5 926

Exercises in Chapter 6 928

Exercises in Chapter 7 929

Exercises in Chapter 8 929

Exercises in Chapter 9 930

Exercises in Chapter 10 931

Exercises in Chapter 11 931

Exercises in Chapter 12 932

Exercises in Chapter 13 932

Index..... 935

About the Authors

Kishori Sharan works as a senior software engineer lead at IndraSoft, Inc. He earned a master of science degree in computer information systems from Troy University, Alabama. He is a Sun-certified Java 2 programmer and has over 20 years of experience in developing enterprise applications and providing training to professional developers using the Java platform.

Peter Späth graduated in 2002 as a physicist and soon afterward became an IT consultant, mainly for Java-related projects. In 2016, he decided to concentrate on writing books on various aspects, but with the main focus set on software development. With two books about graphics and sound processing, three books for Android app development, and several books about Java and Jakarta EE development, the author continues his effort in writing software development-related literature.

About the Technical Reviewers



Massimo Nardone has more than 25 years of experience in security, web/mobile development, cloud, and IT architecture. His true IT passions are security and Android. He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years. He holds a master of science degree in computing science from the University of Salerno, Italy.

He has worked as a CISO, CSO, security executive, IoT executive, project manager, software engineer, research engineer, chief security architect, PCI/SCADA auditor, and senior lead IT security/cloud/SCADA architect for many years. His technical skills include security, Android, cloud, Java, MySQL, Drupal, Cobol, Perl, web and mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, and more.

He worked as a visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas). He is currently working for Cognizant as head of cyber security and CISO to help both internally and externally with clients in areas of information and cyber security, like strategy, planning, processes, policies, procedures, governance, awareness, and so forth. In June 2017, he became a permanent member of the ISACA Finland Board.

Massimo has reviewed more than 45 IT books for different publishing companies and is the co-author of *Pro Spring Security: Securing Spring Framework 5 and Boot 2-based Java Applications* (Apress, 2019), *Beginning EJB in Java EE 8* (Apress, 2018), *Pro JPA 2 in Java EE 8* (Apress, 2018), and *Pro Android Games* (Apress, 2015).

Satej Kumar Sahu works in the role of Senior Enterprise Architect at Honeywell. He is passionate about technology, people, and nature. He believes through technology and conscientious decision making, each of us has the power to make this world a better place. In his free time, he can be found reading books, playing basketball, and having fun with friends and family.

Introduction

How This Book Came About

My first encounter with the Java programming language was during a one-week Java training session in 1997. I did not get a chance to use Java in a project until 1999. I read two Java books and took a Java 2 programmer certification examination. I did very well on the test, scoring 95%. The three questions that I missed on the test made me realize that the books that I had read did not adequately cover details of all the topics. I made up my mind to write a book on the Java programming language. So I formulated a plan to cover most of the topics that a Java developer needs to use Java effectively in a project, as well as to become certified. I initially planned to cover all essential topics in Java in 700–800 pages.

As I progressed, I realized that a book covering most of the Java topics in detail could not be written in 700–800 pages. One chapter alone that covered data types, operators, and statements spanned 90 pages. I was then faced with the question, “Should I shorten the content of the book or include all the details that I think a Java developer needs?” I opted for including all the details in the book, rather than shortening its content to maintain the original number of pages. It has never been my intent to make lots of money from this book. I was never in a hurry to finish this book because that rush could have compromised the quality and coverage. In short, I wrote this book to help the Java community understand and use the Java programming language effectively, without having to read many books on the same subject. I wrote this book with the plan that it would be a comprehensive one-stop reference for everyone who wants to learn and grasp the intricacies of the Java programming language.

One of my high-school teachers used to tell us that if one wanted to understand a building, one must first understand the bricks, steel, and mortar that make up the building. The same logic applies to most of the things that we want to understand in our lives. It certainly applies to an understanding of the Java programming language. If you want to master the Java programming language, you must start by understanding its basic building blocks. I have used this approach throughout this book, endeavoring to build upon each topic by describing the basics first. In the book, you will rarely find a

INTRODUCTION

topic described without first learning about its background. Wherever possible, I tried to correlate the programming practices with activities in daily life. Most of the books about the Java programming language available on the market either do not include any pictures at all or have only a few. I believe in the adage “A picture is worth a thousand words.” To a reader, a picture makes a topic easier to understand and remember. I have included plenty of illustrations in the book to aid readers in understanding and visualizing the concepts. Developers who have little or no programming experience have difficulty in putting things together to make it a complete program. Keeping them in mind, I have included over 390 complete Java programs that are ready to be compiled and run.

I spent countless hours doing research when writing this book. My main sources were the Java Language Specification, whitepapers and articles on Java topics, and Java Specification Requests (JSRs). I also spent quite a bit of time reading the Java source code to learn more about some of the Java topics. Sometimes, it took a few months of researching a topic before I could write the first sentence on it. Finally, it was always fun to play with Java programs, sometimes for hours, to add them to the book.

Introduction to the Second Edition

I am pleased to present the second edition of the *Java Language Features* book. It is the second book in the three-volume “Beginning Java 9” series. It was not possible to include all JDK9 changes in one volume. I have included JDK9-specific changes at appropriate places in the three volumes, including this one. If you are interested in learning only JDK9-specific topics, I suggest you read my *Java 9 Revealed* book (ISBN 9781484225912). There are several changes in this edition, as follows:

- I added the following five chapters to this edition: Implementing Services, The Module API, Breaking Module Encapsulation, Reactive Streams, and Stack Walking.
- Implementing services in Java is not new to JDK9. I felt this book was missing a chapter on this topic. A chapter covers in detail how to define services and service interfaces and how to implement service interfaces using JDK9-specific and pre-JDK9 constructs. This chapter shows you how to use them and provides statements in a module declaration.

- Another chapter covers the Module API in detail, which gives you programmatic access to modules. This chapter also touches on some of the advanced topics, such as module layers. The first volume of this series covered basics on modules, such as how to declare modules and module dependence.
- The following chapter covers how to break module encapsulation using command-line options. When you migrate to JDK9, there will be cases requiring you to read the module's internal APIs or export non-exported packages. You can achieve these tasks using command-line options covered in this chapter.
- Reactive Streams is an initiative for providing a standard for asynchronous stream processing with non-blocking backpressure. It is aimed at solving the problems processing a stream of items, including how to pass a stream of items from a publisher to a subscriber without requiring the publisher to block or the subscriber to have an unbounded buffer. One more chapter covers the Reactive Streams API, which was added in JDK9.
- A new chapter covers the Stack-Walking API, which was added in JDK9. This API lets you inspect the stack frames of threads and get the class reference of the caller class of a method. Inspecting a thread's stack and getting the caller's class name were possible before JDK9. The new Stack-Walking API lets you achieve this easily and efficiently.
- I received several emails from the readers about the fact that the books in this series do not include questions and exercises, which are needed mainly for students and beginners. Students use this series in their Java classes, and many beginners use it to learn Java. Due to this popular demand, I spent over 60 hours preparing questions and exercises at the end of each chapter. My friend Preethi offered her help and provided the solutions.

Apart from these additions, I updated all the chapters that were part of the first edition. I edited the contents to make them flow better, changed or added new examples, and updated the contents to include JDK9-specific features.

It is my sincere hope that this edition will help you learn Java better.

Introduction to the Third Edition

The third edition is the second author Peter Späth's work. Pleasantly taking over much of Kishori Sharan's efforts, the original text was substantially shortened by omitting a couple of chapters, and instead adding API-related topics from the book *Java APIs, Extensions and Libraries*, again from Kishori Sharan. In addition, all topics covered were hovered to Java 17, in order to maximize the benefit for the reader facing contemporary Java projects and wishing to use the new features included with the JRE 17.

Caution Oracle changed the licensing with JDK8. You must enter a paid program if you plan to use Oracle's JRE or JDK for commercial projects. If you want to avoid this, consider using OpenJDK.

Structure of the Book

This book contains 14 chapters. The first seven chapters contain language-level topics of Java such as annotations, reflection, generics, lambda expressions, streams, etc. The chapters introduce Java topics in increasing order of complexity. The subsequent six chapters introduce some of the more important Java APIs and modules, like network programming, remote method invocation, scripting, and more. The last chapter, "Miscellanea," gives the rationale for chapters omitted in this edition compared to the previous one.

In the appendix, solution hints to the exercises are provided.

Audience

This book is designed to be useful to anyone who wants to learn the Java programming language. If you are a beginner, with little or no programming background in Java, you are advised to read one of the beginning-level Java books from Apress, and also the online Java documentation including the Java tutorial will help. This book contains topics of various degrees of complexity. As a beginner, if you find yourself overwhelmed while reading a section in a chapter, you can skip to the next section or the next chapter and revisit it later when you gain more experience.

If you are a Java developer with an intermediate or advanced level of experience, you can jump to a chapter or to a section in a chapter directly. If a section covers an unfamiliar topic, you need to visit that topic before continuing the current one.

If you are reading this book to get a certification in the Java programming language, you need to read almost all of the chapters, paying attention to all of the detailed descriptions and rules. Most of the certification programs test your fundamental knowledge of the language, not the advanced knowledge. You need to read only those topics that are part of your certification test. Compiling and running the Java programs included with the book will help you prepare for your certification.

If you are a student who is attending a class on the Java programming language, you should read the chapters of this book selectively. Some topics, such as lambda expressions, collections, and streams, are used extensively in developing Java applications, whereas other topics are infrequently used. You need to read only those chapters that are covered in your class syllabus. I am sure that you, as a Java student, do not need to read the entire book page by page.

How to Use This Book

This book is the beginning, not the end, of learning the Java programming language. If you are reading this book, it means you are heading in the right direction to learn the Java programming language, which will enable you to excel in your academic and professional career. However, there is always a higher goal for you to achieve, and you must constantly work hard to achieve it. The following quotations from some great thinkers may help you understand the importance of working hard and constantly looking for knowledge with both your eyes and mind open.

The learning and knowledge that we have, is, at the most, but little compared with that of which we are ignorant.

—Plato

True knowledge exists in knowing that you know nothing. And in knowing that you know nothing, that makes you the smartest of all.

—Socrates

INTRODUCTION

Readers are advised to use the API documentation for the Java programming language as much as possible while reading this book. The Java API documentation includes a complete list of everything available in the Java class library. You can download (or view) the Java API documentation from the official website of Oracle Corporation at www.oracle.com.

While you read this book, you need to practice writing Java programs. You can also practice by tweaking the programs provided in the book. It does not help much in your learning process if you just read this book and do not practice writing your own programs. Remember that “practice makes perfect,” which is also true in learning how to program in Java.

Source Code and Errata

Source code for this book can be accessed by clicking the **Download Source Code** button located at www.apress.com/9781484271346.

CHAPTER 1

Annotations

In this chapter, you will learn:

- What annotations are
- How to declare annotations
- How to use annotations
- What meta-annotations are and how to use them
- Commonly used annotations that are used to deprecate APIs, to suppress named compile-time warnings, override methods, and declare functional interfaces
- How to access annotations at runtime
- How to process annotations in source code

All example programs in this chapter are a member of a `jdojo.annotation` module, as declared in Listing 1-1.

Listing 1-1. The Declaration of a `jdojo.annotation` Module

```
// module-info.java
module jdojo.annotation {
    exports com.jdojo.annotation;
}
```

What Are Annotations?

Before I define annotations and discuss their importance in programming, let's look at a simple example. Suppose you have an `Employee` class, which has a method called

`setSalary()` that sets the salary of an employee. The method accepts a parameter of the type `double`. The following snippet of code shows a trivial implementation for the `Employee` class:

```
public class Employee {
    public void setSalary(double salary) {
        System.out.println("Employee.setSalary():" +
            salary);
    }
}
```

A `Manager` class inherits from the `Employee` class. You want to set the salary for managers differently. You decide to override the `setSalary()` method in the `Manager` class. The code for the `Manager` class is as follows:

```
public class Manager extends Employee {
    // Override setSalary() in the Employee class
    public void setSalary(int salary) {
        System.out.println("Manager.setSalary():" +
            salary);
    }
}
```

There is a mistake in the `Manager` class, when you attempt to override the `setSalary()` method. You'll correct the mistake shortly. You have used the `int` data type as the parameter type for the incorrectly overridden method. It is time to set the salary for a manager. The following code is used to accomplish this:

```
Employee ken = new Manager();
int salary = 200;
ken.setSalary(salary);
Employee.setSalary():200.0
```

This snippet of code was expected to call the `setSalary()` method of the `Manager` class, but the output does not show the expected result.

What went wrong in your code? The intention of defining the `setSalary()` method in the `Manager` class was to override the `setSalary()` method of the `Employee` class, not to overload it. You made a mistake. You used the type `int` as the parameter type in the

`setSalary()` method, instead of the type `double` in the `Manager` class. You put comments indicating your intention to override the method in the `Manager` class. However, comments do not stop you from making logical mistakes. You might spend, as every programmer does, hours and hours debugging errors resulting from this kind of logical mistake. Who can help you in such situations? Annotations might help you in a few situations like this.

Let's rewrite your `Manager` class using an annotation. You do not need to know anything about annotations at this point. All you are going to do is add one word to your program. The following code is the modified version of the `Manager` class:

```
public class Manager extends Employee {
    @Override
    public void setSalary(int salary) {
        System.out.println("Manager.setSalary():" +
            salary);
    }
}
```

All you have added is an `@Override` annotation to the `Manager` class and removed the “dumb” comments. Trying to compile the revised `Manager` class results in a compile-time error that points to the use of the `@Override` annotation for the `setSalary()` method of the `Manager` class:

```
Manager.java:2: error: method does not override or
    implement a method from a supertype
    @Override
    ^
1 error
```

The use of the `@Override` annotation did the trick. The `@Override` annotation is used with a non-static method to indicate the programmer's intention to override the method in the superclass. At the source code level, it serves the purpose of documentation. When the compiler comes across the `@Override` annotation, it makes sure that the method really overrides the method in the superclass. If the method annotated does not override a method in the superclass, the compiler generates an error. In your case, the `setSalary(int salary)` method in the `Manager` class does not override any method in the superclass `Employee`. This is the reason that you got the error. You may realize that

using an annotation is as simple as documenting the source code. However, they have compiler support. You can use them to instruct the compiler to enforce some rules. Annotations provide benefits much more than you have seen in this example. Let's go back to the compile-time error. You can fix the error by doing one of the following two things:

- You can remove the `@Override` annotation from the `setSalary(int salary)` method in the `Manager` class. It will make the method an overloaded method, not a method that overrides its superclass method.
- You can change the method signature from `setSalary(int salary)` to `setSalary(double salary)`.

Since you want to override the `setSalary()` method in the `Manager` class, use the second option and modify the `Manager` class as follows:

```
public class Manager extends Employee {
    @Override
    public void setSalary(double salary) {
        System.out.println("Manager.setSalary():" +
            salary);
    }
}
```

Now the following code will work as expected:

```
Employee ken = new Manager();
int salary = 200;
ken.setSalary(salary);
Manager.setSalary():200.0
```

Note that the `@Override` annotation in the `setSalary()` method of the `Manager` class saves you debugging time. Suppose you change the method signature in the `Employee` class. If the changes in the `Employee` class make this method no longer overridden in the `Manager` class, you will get the same error when you compile the `Manager` class again. Are you starting to understand the power of annotations? With this background in mind, let's start digging deep into annotations.