

JAVA ***FOR*** ***ALIENS***

Preface

Java for Aliens is structured in such a way as to facilitate the learning of the Java language (Version 13), even for those who have never programmed, or for those who have programmed with functional languages. The structure, writing and contents have been chosen carefully, based on the experience I have accumulated as a trainer and mentor over a twenty-year career. In particular, for Sun Microsystems, for Oracle, and as a freelancer, I've had the opportunity to deliver about 250 courses, for thousands of learners, on Java technology, architecture, object-oriented analysis, design and UML. This book also covers complex topics that even the most experienced Java programmers might not master. These topics are marked with appropriate icons to warn the reader about the frequency of use, complexity and importance.

This two-volume book represents an evolution of the most popular book on Java in Italy, which since 2006, has been recommended as a reference book in all the most important Italian universities. The “Java Manual”, (published by Hoepli), boasts the highest number of reviews on Amazon Italy compared to all other programming books, even those of the most established international authors. Furthermore, my free book on Java 5 (<http://www.claudiodesio.com>), has been downloaded over 450,000 times out of a total of about 2 million Italian programmers of all languages.

The book you have in your hands is therefore the result of years of work, based not only on practical and educational experience, but also on extensive feedback from readers.

What's New?

From Version 9 of Java onwards, a new six-monthly release cycle has been introduced. This book is being published at the same time as Version 13, and also covers all of the new features in Versions 9, 10, 11 and 12, often with a very high level of detail. Each topic introduced in these versions will be presented with a specific icon that is clearly identifiable. After the Java 8 revolution, Versions 9 to 13 have brought new constructs (switch expressions, text boxes, modules...), new ways to launch applications (JShell, Launch Single Source Files...), new Garbage Collectors (G1, Epsilon, ZGC, Shenandoah...), new syntax words (var, opens, exports...), new application types (Modules, Custom Runtime Images, Modular Jar...), new JDK tools (jlink, jmod...) and much more ... including loads of challenging insights for more skilled aliens!

Book Structure

Java for Aliens is divided into 7 parts within two volumes:

1. Part I, “Java Language Basics” (Chapters 1 to 5 of Volume 1), presents all the fundamental concepts of Java programming, such as the development environment, essential components, programming constructs (including novelties such as the `switch` expression), data types, etc. The study of the first five chapters should therefore allow the reader to write their first programs, and also to be confident in the development environment. Nothing will be taken for granted, and notions of basic computer science will also be introduced. Furthermore, the reader will receive support in the form of simplified examples, exercises and explanations. However, there will also be an in-depth analysis, especially on the latest new features. I have also created a simple, free and open source development tool called EJE, to support the initial learning phases.
2. Part II, “Object Orientation” (Chapters 6 to 10 of Volume 1), explains the fundamental concepts for correctly designing our programs. How do we organize a program from scratch? How many classes do we need to create? What should these classes be called? What methods will they have to define? How can we create a program capable of evolving without changing parts that have already been written? These and many other questions are answered by the theory of Object Orientation with its paradigms (abstraction, reuse, encapsulation, inheritance, polymorphism and then cohesion and coupling, and the principle of inversion of dependence, etc.). In this part, particular emphasis is placed on the support that the language offers to Object Orientation. Usually, the major difficulty that a Java programmer faces is being able to exploit the paradigms of object-oriented programming in practice. This text then strives to provide all the information needed by the reader in order to follow the path of Java programming in the most correct way possible, that is, the object-oriented way. Further, important features such as abstract classes, interfaces, packages, initializers, design by contract with assertions, handling exceptions, errors and warnings, etc. will be presented contextually.
3. Part III, “Java Language Advanced Features” (Chapters 11 and 12 of Volume 1), introduces some more complex topics such as enumerations, generic types, erasure, wildcards, bounded wildcards, bounded parameters, generic methods, intersection types, wildcard capture, helper methods, covariant parameters, nested types, and anonymous classes. These arguments are in preparation for those that will be presented in the chapters that follow. Furthermore, at this point and in other chapters, the discussion is extensively developed with a view to supporting possible Oracle certification (both OCA and OCP).

4. Part IV, “Java API Fundamentals” (Chapters 13 to 15 of Volume 2), presents the fundamental standards libraries for language and utilities. These include the *Date & Time API*, the *Reflection API* and all the fundamental classes such as `Object`, `System`, `String`, `Runtime`, `Math`, etc. Further, it introduces APIs for managing string formatting, internationalizing our applications, creating configuration files, using regular expressions. It also explains how to manage *concurrent (multi-threaded) programming* and the related libraries that support it.
5. Part V, “Java Language Evolution” (Chapters 16 to 19 of Volume 2), presents the main features introduced in recent years that have revolutionized the language, such as *functional programming* with lambda expressions and method references, the *Fluent API* with streams, *meta-programming* with annotations, and the Java Platform Module System that has changed the way we design Java programs today - a key part for keeping up with the times.
6. Part VI, “Java Integration API” (Chapters 20 to 22 of Volume 2), introduces the Java support structures that allow us to interact with other technologies or systems. The Java Native Interface, the `java.io`, `java.nio` packages (and the sub-packages that define *NIO 2 API*), networking support with `java.net`, the JDBC interface to connect with databases, and libraries for interacting with programs and technologies based on XML, are featured in this part of the book.
7. Finally, Part VII, “Java Graphical User Interfaces” (Chapters 23 and 24 of Volume 2), presents the libraries that will allow us to create graphical user interfaces with Java. In particular, we will learn to use the AWT, Swing and JavaFX libraries, with a view to creating any kind of user interface for standalone programs.

This is therefore a book that should satisfy the expectations of both the aspiring programmer and the expert programmer.

Moreover, in order not to burden the body and the cost of the work too much, a lot of material (including over 500 exercises) has been moved to a special space available online at this address: <http://www.javaforaliens.com>.

Text Styles

The text is stylized to capture attention, highlighting words, sentences or entire paragraphs. In particular: with the *Italic* style, particular or important terms are highlighted; the **Bold** style highlights key words, important concepts of the Java language, names of technologies or other things the author considered important and sought to draw attention to; the **Interface** style is used, within the text, for links, commands and everything that can be typed in the interfaces of

an operating system; the Code in the text style is used to distinguish parts of the code in the explanations from the rest of the text. The code lines are inserted in highlighted blocks similar to the following:

```
public class MyFirstJava13Class {  
    //This is Java 13 code!  
}
```

while the command line inputs and execution outputs are formatted in blocks with a black background:

```
this is the output of my program
```

Finally, notes like this add information to the topic before or after it. They can be insights, suggestions, or even simple references to other parts of the work.

Semantic Icons

To help visually identify parts of the text with particular semantic characteristics, the following icons have been used.

New Feature
in Java 9



New Feature
in Java 10



New Feature
in Java 11



New Feature
in Java 12



New Feature
in Java 13



Curiosity
(Non-essential
concept or story)



Trick



Common Error



Rarely Used



Best Practice



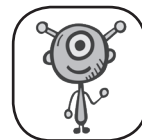
Very Important



Complex Concept



Only for Aliens



We recommend the novice reader not to dwell on the parts marked with the icon “Only for Aliens” and “Rarely Used”, in case they are too complicated. These are not essential parts for language learning, and you can go back to reading them when you feel ready.

Who Should Read This Book?

Java for Aliens has been structured to meet the expectations of:

- Aspiring programmers who want to become pro: nothing is taken for granted, you can learn to program even from scratch, and enter the world of work from the front door.
- University students: the previous versions of this work have been adopted as a textbook for many university courses in all the major Italian universities.
- Expert Java programmers: who need to upgrade to Version 13 and keep up with the times.

Online Resources

A lot of resources have been moved online for this edition. This could easily have given rise to a third volume! In particular, in addition to fifteen appendices, the errata, all the source code of the examples, there are about 500 exercises (and their solutions), fundamental for learning the concepts presented in the paper book.

You can download all the exercises and the appendices and the other files in a special space available online accessible at <http://www.javaforaliens.com>.

You can also subscribe to the Telegram channel <https://t.me/java4aliens> in order to be notified about future updates of the online material.

Source Code

Working with the code examples in this book, you can decide to write the whole code by hand, or to use the source files that come with the book. All of the source code written in this book is available for download at <http://www.javaforaliens.com>.

Errata

Every effort has been made to avoid errors in the text and in the code.

However, nobody is perfect, and mistakes happen. If you find a spelling error or a part of the code that is not working, the author would be grateful for the feedback and it will be reported in a new version of the errata. Other readers will thus avoid hours of frustration and, at the same time, it will help to ensure the highest possible standards in terms of quality of information. You can do this by writing to the author at claudio@javaforaliens.com. As usual, the reference address remains: <http://www.javaforaliens.com>.

Author

Since 1999, I have worked as a freelance IT consultant. Today I am a specialist in training, technical writing, development, analysis, design, Java technologies, architecture and object-oriented methodologies. I am the author of several technical articles and the “Manuale di Java” series from Version 6 to 9, Italian bestsellers, all published by Hoepli. I have worked with several universities, ministerial authorities and IT companies including Sun Microsystems, as a trainer and mentor. Today, I mainly work as a training consultant for Oracle.

Contacts

You can contact me through the following channels:

- **Internet:** <http://www.claudiodesio.com>
- **E-mail:** claudio@claudiodesio.com
- **Facebook:** <http://www.facebook.com/claudiodesiocesari>
- **Twitter:** <http://twitter.com/cdesio>
- **LinkedIn:** <http://www.linkedin.com/in/claudiodesio>
- **YouTube:** <https://www.youtube.com/c/claudiodesiocesari>

Acknowledgments

My thanks go to all of the people who have supported me throughout this work, and to all those who, in some way, have contributed to its realization.

A special thanks goes to my editor Emanuele Giuliani (emanuele@giuliani.mi.it) and my English guru, 'Proofreading Guy' (<http://www.proofreadingguy.com>) who has taken care of this work with great patience and professionalism.

I would also like to thank (though it's never enough) my parents, for all they have done for me and continue to do, and for their unconditional love.

Finally, I thank my family, who are my reason for living: Rosalia, Andrea and Simone. This book, as always, is dedicated to you. You are everything to me.



Claudio De Sio Cesari

To Rosalia, Andrea and Simone...

CLAUDIO DE SIO CESARI

JAVA FOR ALIENS

**LEARN JAVA FROM SCRATCH
AND BECOME A PRO**

Volume 1

Java for Aliens - Volume 1

Copyright © 2019 by **Claudio De Sio Cesari**

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, without the prior written permission of the author, except in the case of brief quotations permitted by copyright law. For permission requests, write to the author at the address: **claudio@claudiodesio.com**

Proofreader: Proofreading Guy (<http://www.proofreadingguy.com>)

Book Editor and Cover Designer: Emanuele Giuliani (emanuele@giuliani.mi.it)

Cover Designers: Ciro Improta (improta@libero.it)

Riccardo Ferti (riccardo.ferti@gmail.com)

Andrea Massaretti (andrea.massaretti@gmail.com)

Printed by: kdp.amazon.com

First Edition (November, 2019)

ISBN: 979-12-200-4915-3

Font licenses

Libre Baskerville (<https://fonts.google.com/specimen/Libre+Baskerville>, Impallari Type): OFL

Libre Franklin (<https://fonts.google.com/specimen/Libre+Franklin>, Impallari Type): OFL

Cousine (<https://fonts.google.com/specimen/Cousine>, Steve Matteson): AL

Inconsolata (<https://fonts.google.com/specimen/Inconsolata>, Raph Levien): OFL

Roboto (<https://fonts.google.com/specimen/Roboto>, Christian Robertson): AL

Digits (<https://www.1001fonts.com/digits-font.html>, Dieter Steffmann): FFC

Journal Dingbats 3 (<https://www.1001fonts.com/journal-dingbats-3-font.html>, Dieter Steffmann): FFC

Musicals (<https://www.1001fonts.com/musicals-font.html>, Brain Eaters): FFC

Image licenses

Curiosity icon (https://www.flaticon.com/free-icon/toyger-cat_107975, www.freepik.com): FBL

Alien icon (<http://www.iconarchive.com/show/free-space-icons-by-goodstuff-no-nonsense/alien-4-icon.html>, goodstuffnononsense.com): CC

Trick icon (https://www.flaticon.com/free-icon/magic-wand_1275106, www.flaticon.com/authors/pause08): FBL

Millennium Falcon image: (<https://free-clipart-pictures.com/startdownload.html?id=951327>): FCP.com

License specifications

Open Free License (OFL): https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL_web

Apache License, Version 2.0 (AL): <http://www.apache.org/licenses/LICENSE-2.0>

1001Fonts Free For Commercial Use License (FFC): <https://www.1001fonts.com/licenses/ffc.html>

Flaticon Basic License (FBL): <https://file000.flaticon.com/downloads/license/license.pdf>

CC Attribution 4.0 (CC): <https://creativecommons.org/licenses/by/4.0/legalcode>

Free-clipart-pictures.com (FCP.com): <https://free-clipart-pictures.com/terms.html>

Any other trademarks, service marks, product names or named features are assumed to be the property of their respective owners, and are used only for reference. There is no implied endorsement if we use one of these terms.

Table of Contents

Preface

XVII

Part I - Java Language Basics 1

Chapter 1 - Introduction to Java 3

1.1 Prerequisites: Basic IT Concepts	4
1.1.1 Hardware	4
1.1.2 Software	5
1.1.3 Programming Languages	5
1.2 Introduction to Java	7
1.2.1 What is Java?	8
1.2.2 Java Language Features	8
1.2.3 False Beliefs About Java	11
1.3 Development Environment	12
1.3.1 Structure of the JDK	13
1.3.2 Step by Step Developing Guide	15
1.4 First Approach to the Code	16
1.4.1 Analysis of the HelloWorld Program	17
1.4.2 Compiling and Running the HelloWorld Program	20
1.4.3 Our Interaction with the Computer	21
1.5 Problems That Can Occur During the Compilation and Execution Phases	21
1.5.1 Problems That Can Occur During the Compilation Phase	22
1.5.2 Problems That Can Occur During the Execution Phase	23
1.6 Other Development Environments	23
Summary	25

Chapter 2 - Key Components of a Java Program 27

2.1 What It Means to Create a Java Program	28
2.1.1 Programming and paradigms	29
2.1.2 Object-Oriented Programming (OOP)	30

2.2 The Basics of Object-Oriented Programming: Classes and Objects	31
2.2.1 About the Point Class	35
2.2.2 About the PointObjectsTest Class	37
2.2.3 About Objects (Again)	37
2.3 Java Methods	39
2.3.1 Method Declaration	39
2.3.2 Calling (invoking) a Method	41
2.3.3 Varargs	43
2.4 Variables in Java	44
2.4.1 Variable Declaration	44
2.4.2 Instance Variable	45
2.4.3 Local Variables	46
2.4.4 Parameters	47
2.5 Constructors	48
2.5.1 Constructor Features	49
2.5.2 Default Constructor	51
2.6 Introduction to Packages	53
2.6.1 Package and Import	53
2.6.2 Manual Package Management	55
2.7 Other Key Components of Java Programming	57
2.7.1 Interfaces	57
2.7.2 Enumerations	59
2.7.3 Annotations	59
2.7.4 Modules	60
2.7.5 Initializers	61
2.7.6 Nested Classes	61
2.7.7 Lambda Expressions	61
Summary	62
 Chapter 3 - Coding Style, Data Types and Arrays	 65
3.1 Coding Style	66
3.1.1 Free-Form	66
3.1.2 Comments	68
3.1.3 Rules for Identifiers	70
3.1.4 Standards and Naming Conventions	72
3.2 Memory Management	74
3.2.1 Bytes	74
3.2.2 Numeral Systems	76
3.2.3 Two's Complement	76

3.3 Primitive Data Types	77
3.3.1 Integer Data Types	79
3.3.1.1 Automatic Promotion in Expressions	80
3.3.1.2 Type Casting	81
3.3.1.3 Operations with Integers	82
3.3.2 Floating-Point Data Types, Cast and Promotion	83
3.3.2.1 Floating Point Numbers Precision (IEEE-754)	85
3.3.3 Underscore in Numeric Data Types	87
3.3.4 Boolean-Logical Data Type	87
3.3.5 Primitive Character Data Type	88
3.3.5.1 Printable Keyboard Characters	89
3.3.5.2 Unicode Format (Hexadecimal Notation)	90
3.3.5.3 Special Escape Characters	91
3.3.5.4 Write Java Code with the Unicode Format	93
3.3.5.5 Unicode Format for Escape Characters	93
3.3.5.6 Unicode Format and Comments	96
3.3.5.7 Casting Between short and char	96
3.4 Non-Primitive Data Types: Reference	97
3.4.1 Passing Parameters by Value	99
3.4.2 The Initialization of Instance Variables	102
3.5 Introduction to the Standard Library (Java API)	102
3.5.1 The <code>import</code> Command	103
3.5.2 The <code>String</code> class	105
3.5.3 Java API Documentation	106
3.5.4 The <code>javadoc</code> Tool	107
3.6 Arrays in Java	108
3.6.1 Declaration	109
3.6.2 Creation	109
3.6.3 Initialization	109
3.6.4 Alternative Syntaxes	110
3.6.5 Characteristics of an Array	110
3.6.6 Multidimensional Arrays	111
3.6.7 String <code>args[]</code>	112
3.7 Local Variables Type Inference: <code>var</code>	113
3.7.1 Verbosity and readability	113
3.7.2 Applicability	115
3.7.3 Others uses of <code>var</code>	116
3.7.4 The <code>var</code> Reserved Type Name	116
Summary	117

Chapter 4 - Operators and Execution Flow Management	121
4.1 Basic Operators	121
4.1.1 Assignment Operator	122
4.1.2 Arithmetic Operators	122
4.1.2.1 Compound Assignment Operators	123
4.1.2.2 Pre and Post-Increment (and Decrement) Unary Operators	124
4.1.3 Bitwise Operators	126
4.1.3.1 The NOT Operator ~	127
4.1.3.2 Bitwise Logic Operators	127
4.1.3.3 Shift Operators	127
4.1.4 Relational or Comparison Operators	129
4.1.5 Boolean - Logical Operators	130
4.1.6 String Concatenation with +	131
4.1.7 Operator Priority	132
4.2 Simple Programming Constructs	133
4.2.1 The if Construct	134
4.2.2 The while Construct	137
4.3 Advanced Programming Constructs	138
4.3.1 The for Construct	138
4.3.1.1 for vs while	140
4.3.2 The do Construct	141
4.3.3 Enhanced for Loop	142
4.3.4 The switch Construct	144
4.3.4.1 Syntax	144
4.3.4.2 Fallthrough	146
4.3.4.3 How to Use the switch Construct	148
4.3.5 The Ternary Operator	149
4.3.6 Support for Programming Constructs break and continue	150
4.3.6.1 The break Keyword	150
4.3.6.2 The continue Keyword	151
4.3.6.3 Labels	151
4.4 Experimental Programming Construct (Feature Preview)	152
4.4.1 The switch Expression	153
4.4.2 Syntax and Arrow Notation	153
4.4.3 Feature Preview	155
4.4.4 Arrow vs Colon	156
4.4.5 Poly Expression	157
4.4.5.1 First Scenario (Explicit Type)	157
4.4.5.2 Second Scenario (var)	158
4.4.6 Switch as a Statement	159
4.4.7 Exhaustiveness	161

4.5 Better Programming	163
4.5.1 Approach to Programming	164
4.5.2 Algorithms	164
4.5.3 Introduction to UML	166
Summary	169
 Chapter 5 - Real Development with Java	 171
<hr/>	
5.1 Executing a Program Using the Source File (Without Compiling)	172
5.1.1 When to Use This Technique	173
5.1.2 Launching Files Instead of Classes	174
5.1.3 Shebang Files	176
5.2 JShell	177
5.2.1 HelloWorld with JShell	177
5.2.2 JShell and Java Code	179
5.2.2.1 Practical Examples	180
5.2.2.2 JShell Rules	182
5.2.3 JShell Commands	183
5.2.3.1 Exploratory Commands	183
5.2.3.2 Executive Commands	185
5.2.4 JShell Auxiliary Tools	187
5.2.4.1 Implicit Variables	187
5.2.4.2 Forwarding Reference	188
5.2.4.3 Auto-Completion	188
5.2.4.4 The /edit Command	189
5.2.4.5 Keyboard Shortcuts	190
5.3 Integrated Development Environment (IDE)	191
5.3.1 Project	192
5.3.2 Editor	192
5.3.3 Debugger	192
5.3.4 Integrations	193
5.4 Business Reality	193
5.4.1 Architecture	194
5.4.2 Knowledge	195
5.4.3 Types of Programs	196
5.4.4 Roles	197
5.4.5 Methodology	198
5.5 Case Study	199
5.5.1 Requirements Gathering with Use Cases and Scenarios	199
5.5.1.1 Use Case Diagram	200
5.5.1.2 Use Case Scenarios	201

5.5.2 High Level Architecture	203
5.5.3 Key Abstractions	203
5.5.4 Design with Interaction Diagrams	204
Summary	205
Part I Conclusions	207

Part II - Object Orientation 209

Chapter 6 - Encapsulation and Scope 211

6.1 Brief History of Object-Oriented Programming	212
6.2 Object-Oriented Paradigms	214
6.2.1 Abstraction	215
6.2.2 Reuse	216
6.2.3 UML: Class Diagram	217
6.3 Encapsulation	218
6.3.1 Functional Encapsulation	224
6.3.2 Encapsulation and Reuse	225
6.3.3 The this Reference	227
6.3.3.1 Using this with Variables	228
6.3.3.2 Use of this with Methods	230
6.3.3.3 Use of this with Constructors	231
6.3.3.4 this and assignment	232
6.4 When to Use Encapsulation	233
6.5 How to Use Encapsulation	235
6.5.1 IDEs	235
6.5.2 Other Forms of Encapsulation	236
6.6 Package Management	237
6.6.1 Manual Management	239
6.6.2 Modules	241
6.7 Access Modifiers	241
6.7.1 The public Modifier	241
6.7.2 The protected Modifier	242
6.7.3 No Access Modifier	245
6.7.4 The private Modifier	246
6.8 The static Modifier	246
6.8.1 Static Methods	247
6.8.2 Static Variables (Class Variables)	248
6.8.3 Static Initializers and Object Initializers	251
6.8.3.1 When a class is loaded into memory	252
6.8.3.2 Object Initializers	253

6.8.4 Static Import	253
6.8.5 When to Use static	255
6.8.6 Singleton Design Pattern	257
Summary	260
 Chapter 7 - Inheritance and Interfaces	 263
<hr/>	
7.1 Inheritance	264
7.1.1 The Keyword extends	264
7.1.2 The Object Class	265
7.1.3 When to Use Inheritance: “Is a” Relationship	266
7.1.4 How to Use Inheritance: Generalization and Specialization	267
7.2 Inheritance Language Support	269
7.2.1 Inheritance-Encapsulation Relationship	269
7.2.2 Inheritance and Constructors	270
7.2.3 The super Keyword	272
7.2.3.1 super and Constructors	272
7.2.3.2 super and Methods	275
7.2.4 Inheritance and Initializers	276
7.2.5 Inheritance and Modifiers	278
7.2.5.1 Access Modifiers	278
7.2.5.2 The static Modifier	280
7.2.5.3 The final Modifier	281
7.3 The abstract Modifier	282
7.3.1 Abstract Methods	282
7.3.2 Abstract Classes	283
7.4 Interfaces	286
7.4.1 Classical Definition (pre-Java 8)	286
7.4.2 Static Methods	288
7.4.3 Default Methods	290
7.4.4 Functional Interfaces	291
7.4.5 Multiple Inheritance	292
7.4.5.1 Diamond Problem	293
7.4.5.2 Conflict Between an Abstract and a Concrete Method	295
7.4.5.3 Conflict Between Two Abstract Methods	296
7.4.5.4 Conflict Between Redefined Methods	296
7.4.5.5 Class Always Wins	297
7.4.5.6 Multiple inheritance and static methods	297
7.4.5.7 Multiple Inheritance and Constants	299
7.4.6 Differences Between Interfaces and Abstract Classes	300
Summary	301

Chapter 8 - Polymorphism	305
8.1 Polymorphism	305
8.1.1 Reference Explained	306
8.2 Polymorphism for Methods	307
8.2.1 Overload	307
8.2.1.1 Method Overloading	308
8.2.1.2 Constructor Overloading	310
8.2.1.3 Overload Ambiguity	311
8.2.2 Varargs	312
8.2.2.1 Varargs vs Arrays	313
8.2.2.2 Varargs vs Overload	314
8.2.3 Override	315
8.2.3.1 Override Rules	317
8.2.3.2 Override Annotation	318
8.2.3.3 Override and static	319
8.3 Polymorphism for Data	321
8.3.1 Polymorphic Parameters	322
8.3.2 Heterogeneous Collections	323
8.3.3 instanceof operator	325
8.3.4 Objects Casting	328
8.3.5 Virtual Method Invocation	330
8.3.6 Example of Use of Polymorphism	331
8.3.7 Polymorphism and Interfaces	334
8.3.8 Polymorphism and Constructors	335
Summary	337
Chapter 9 - Exceptions and Assertions	339
9.1 Exceptions, Errors and Assertions	339
9.2 Throwable Hierarchy	340
9.2.1 Checked and Unchecked Exceptions	341
9.2.2 Frequent Exceptions	342
9.2.3 Custom Exceptions	342
9.3 Exception Handling Mechanism	343
9.3.1 try - catch Blocks	343
9.3.1.1 Exceptions as Information Containers	345
9.3.2 Catching Exceptions	346
9.3.2.1 Handle more exceptions	346
9.3.2.2 Multi- catch	347
9.3.3 The finally Clause	348
9.3.3.1 Closing Closeable Objects	349

9.3.4 Try-with-Resources	351
9.3.4.1 Closeable Objects	352
9.3.4.2 Effectively Final Variables	352
9.3.4.3 Suppressed Exceptions	353
9.3.5 Exception Propagation	356
9.3.5.1 Launching Exceptions: the throw keyword	356
9.3.5.2 Exception Propagation	358
9.3.5.3 The throws keyword	359
9.3.5.4 Overriding Methods: Another Rule	362
9.3.5.5 Warnings	363
9.3.6 More on Exception Handling	365
9.3.6.1 More on entering a finally block	365
9.3.6.2 Exceptions Caused by Other Exceptions	366
9.3.6.3 Loading Classes: ExceptionInInitializerError	368
9.4 Introduction to Assertions	370
9.4.1 Design by Contract	372
9.4.2 Notes for Running Programs that Use the Word assert	372
9.4.3 When to Use Assertions	374
9.4.3.1 First Piece of Advice	374
9.4.3.2 Second Piece of Advice	375
9.4.3.3 Third Piece of Advice	376
9.4.4 Conclusions	379
Summary	380
 Chapter 10 - A Guided Example to Object-Oriented Programming	 383
<hr/>	
10.1 Why This Chapter?	384
10.2 Case Study	385
10.3 Case Study Resolution	385
10.3.1 Step 1	385
10.3.2 Step 2	387
10.3.3 Step 3	388
10.3.4 Step 4	391
10.3.5 Step 5	392
10.4 Introduction to Testing and Logging	393
10.4.1 Unit Testing in Theory	394
10.4.2 Unit Test in Practice with JUnit	395
10.4.3 Java Logging	400
Summary	403
Part II Conclusions	404

Part III - Java Language Advanced Features 405

Chapter 11 - Enumerations and Nested Types 407

11.1 Nested Classes	408
11.1.1 Nested Classes: Definition	408
11.1.2 Nested Classes: Properties	409
11.1.2.1 Access Modifiers and Scope	410
11.1.2.2 Instantiating Nested Classes	410
11.1.2.3 The this Reference	412
11.1.2.4 Local Classes	413
11.1.2.5 Static Nested Classes	415
11.1.2.6 Abstract Nested Classes	416
11.1.2.7 Nested Classes Extended Outside the Containing Class	416
11.1.2.8 Deeper Nested Classes	417
11.1.3 When to Use Nested Classes	418
11.2 Anonymous Classes: Definition	420
11.2.1 Syntax	420
11.2.2 Non-Denotable Type	421
11.2.3 Anonymous Classes vs Inner Classes	422
11.2.3.1 Anonymous Classes and Modifiers	422
11.2.3.2 Properties in Common with the Inner Classes	423
11.2.3.3 Anonymous Classes and Inheritance	424
11.2.4 Anonymous Local Classes and var	425
11.2.5 When to Use Anonymous Classes	426
11.3 Enumeration Types	431
11.3.1 Inheritance and Enumerations	432
11.3.2 Polymorphism and Enumerations	433
11.3.3 Methods, Variables, Constructors and Nested Types within an Enumeration	434
11.3.4 When to Use an enum	437
11.3.5 Nested Enumerations (in Classes) or Member Enumerations	438
11.3.6 Enumerations and Specific Methods of the Elements	440
11.3.7 Switch and enum	441
11.4 More on static imports	442
11.4.1 Using static Imports	444
11.4.1.1 Enumerations	444
11.4.1.2 Abstraction	445
11.4.2 Impact on Java	447
11.4.2.1 Ambiguity	447
11.4.2.2 Shadowing	448
Summary	449

Chapter 12 - Generic Types	451
12.1 Generics and Type Parameters	452
12.1.1 Generics and Collections	454
12.1.2 Primitive Types and Autoboxing-Autounboxing	457
12.1.3 The Iterator Interface	458
12.1.4 The Map Interface	459
12.1.5 Automatic Type Inference	460
12.1.6 Raw Types and XLint	462
12.2 Inheritance and Generic Types	465
12.2.1 Inheritance and Type Parameter	465
12.2.2 Type Erasure	466
12.2.3 Wildcard	467
12.3 Create Our Own Generic Types	469
12.3.1 Bounded Parameters	470
12.3.2 Bounded Wildcard	471
12.3.3 Generic Methods	473
12.3.4 Intersection types	474
12.3.4.1 Example	475
12.3.4.2 Multiple Intersection	477
12.3.4.3 Assignment of Intersection Types	477
12.3.5 Wildcard Capture	478
12.3.5.1 Helper Methods	479
12.4 Covariant Parameters	480
12.4.1 Covariant Parameters Problem	481
12.4.2 Bounded Parameter Solution	483
Summary	485
Part III Conclusions	487
 Java Version New Feature Index	 489
 Word Index	 491

Java Version New Feature Index

List of pages where new features are present for any Java version.



13, 60, 70, 71, 102, 106, 177, 241, 290, 291, 352, 461



71, 113, 139, 144, 412, 425, 478



14, 90, 172



152, 155, 175, 357, 442



71, 90, 152, 154

Word Index

Symbols

! (NOT) 129–130, 132
!= (different from) 129, 130, 132
% 122, 132
%= (modulo and assignment) 123, 132
& (AND) 126, 130, 132
&& (AND) 126, 130–132
&= (AND and assignment) 131, 132
* 122, 132
*= (multiplication and assignment) 123, 132
+ 122, 132
++ (pre-increment of a unit) 124–126, 132
+= (sum and setting) 123–124, 132
- 122, 132
-- (pre-decrease of one unit) 124–126, 132
-= (subtraction and setting) 123–124, 132
-> (used for switch and lambda) 62, 153–162, 357
-disableassertions 372
-enableassertions 372
-Xlint 157, 365, 462–464, 467, 486
/ 122, 132
/= (division and assignment) 123–124, 132
< (lower than) 129, 132
<< (left shift) 126, 128, 132, 133
<=< (left shift and assignment) 126, 128, 132, 133, 169
<= (lower than or equal to) 129, 132
== (equal to) 129–132
> (greater than) 129, 131, 132
>= (greater than or equal to) 129, 132
>> (shift to the right) 126–128, 132
>>= (shift to the right and assignment) 126, 128, 132, 133

>>> (shift to the right without sign) 126, 128, 132
>>>= (shift to the right without sign and assignment) 126, 128, 132, 133
@interface (keyword) 60, 71, 72
\n 78, 88, 91, 93, 96, 276
\r 91, 94
\t 91
^ (XOR) 126, 131, 132
^= (XOR and assignment) 126, 127, 131, 132
| (OR) 126, 130, 132
|| (OR) 126, 130–132
|= (OR and assignment) 126, 127, 131–132
||= (OR and assignment) 126, 127, 131–132
~ (NOT) 126, 127, 132

A

abstract (keyword) 70, 182, 206, 263, 269, 282–288, 290–292, 295–301, 330–335, 356, 386, 416, 423, 425, 428, 430, 431, 433, 434, 458, 472, 475, 476, 484
Abstraction 214, 215, 260, 445
algorithm 164–168, 170, 216, 217, 388, 392, 430
Annotation 318
args[] 17–19, 22, 23, 113, 172, 176, 392, 393
Array 110
Arrays (class) 108, 111, 313
assert (keyword) 31, 70, 212, 252, 340, 370–379, 397, 442
Autoboxing 458
Automatic Type Inference 460
Autounboxing 457

B

bitwise 124, 126, 127, 131
 boolean (keyword) 70, 78, 87, 102, 115, 118, 127,
 129–131, 134–139, 141–143, 145, 150, 180,
 183, 184, 221, 309, 370, 374, 376, 379, 430,
 431, 455, 458, 461, 469
 break (keyword) 70, 113, 138, 144–156, 377, 378,
 437, 441, 442

C

Cast 83
 Class (class) 22, 23, 28, 35, 37, 217, 222, 225,
 226, 230, 232, 248, 253, 258, 265, 268, 270,
 283, 297, 379, 416
 ClassCastException 329, 342, 354–356, 366,
 452, 453, 456, 485
 clone() 111
 Collection (interface) 8, 446, 447, 454, 459
 Collections (class) 111, 142, 323–325, 405,
 454, 457, 458, 465, 486, 487
 Constructor 28, 49, 51, 253, 310, 385, 386
 Conventions 72

D

Decorator (pattern) 112, 391
 Deprecated (annotation) 464
 Design by contract 372, 379
 Diamond Problem 293–296

E

Eclipse 5, 10, 24, 191–193, 235, 239, 240, 260, 363,
 364, 394, 396, 398, 399, 428, 442
 EJE 24, 26, 55, 67, 104, 107, 108, 113, 149, 156,
 196, 221, 235, 239, 259, 270, 363, 365
 Enhanced for Loop 142
 enum (keyword) 22, 23, 59, 70, 159, 162, 431–441,
 444
 Enumeration 431, 434
 equals() 111

Error 23, 32, 87, 179, 181, 281, 340, 342, 380, 466,
 471, 478, 480, 484, 485
 exports (keyword) 60, 70, 71, 464
 extends (keyword) 70, 242, 243, 263–301,
 305–336, 343, 354, 356, 361–364, 386, 389,
 390, 416, 417, 422, 425, 431, 433, 454,
 461–463, 471–473, 475–477, 480, 482, 484

F

Factory Method (pattern) 375

G

Garbage Collector 9, 351
 Generalization 267
 Generics 452, 454, 457, 458

H

HashMap (class) 459–461
 HelloWorld 17, 18, 20–23, 28, 29, 109, 110, 177,
 186, 421
 helper 405, 451, 468, 479, 486, 487
 Heterogeneous Collections 323

I

immutable 106, 118
 implements (keyword) 12, 58, 70, 252, 287–290,
 292–299, 301, 325, 327, 334, 352, 364, 375,
 434, 446, 456, 475–477, 481–484
 import (keyword) 27, 54, 56, 57, 63, 64, 66, 70,
 71, 103, 104, 119, 147, 153, 184–186, 188,
 189, 206, 238, 242, 244, 245, 253, 254, 261,
 279, 280, 396, 397, 400, 439, 442–450
 inheritance 10, 58, 59, 209, 212, 214, 216, 242, 245,
 260, 263–301, 305–336, 372, 384–386, 422,
 428, 451, 465–467, 480, 486, 487
 Initializers 61, 63, 251, 253, 276, 278, 302
 instanceof (keyword) 70, 132, 305, 325–329,
 337, 338, 389, 390, 453, 455, 482
 Iterator XV, 457–459, 462, 463, 467, 468, 472, 473

J

JAR 187
 Javadoc 69, 70, 117, 119
 Java documentation 119, 374
 JShell 2, 24, 78, 171, 177–190, 192, 206, 208, 248,
 466, 478

L

List (interface) 184, 325, 327, 407, 445, 450,
 454–458, 461–463, 465, 466, 469, 470, 472,
 473, 478, 479, 489
 Literals 78

M

Map (interface) 457, 459–461
 Math (class) 247, 248, 250, 254, 255, 281, 290,
 315, 316, 388–390, 443, 448, 449
 Method 28, 39, 41, 72, 285, 291, 295, 308, 330,
 375, 401, 428
 module (keyword) 60, 70, 71, 172, 203, 241, 355,
 400, 464

N

NaN 84
 native (keyword) 9, 70, 197, 278
 NEGATIVE_INFINITY 84
 Netbeans 5, 10, 24, 191, 192, 235, 239, 240, 260,
 363, 364, 394, 428

O

Object (class) 28, 31, 111, 265, 266, 272, 276,
 311, 323, 330, 375, 426, 452, 453, 455, 459,
 462, 463, 466–468, 472, 478, 479
 Overload 307, 309, 311, 314
 Override 276, 285, 315–336, 343, 352, 354, 356,
 363, 389, 390, 416, 420, 423–425, 427,
 434–436, 440, 461, 469, 475, 480

P

Package 28, 53, 55, 73, 237, 238
 Polymorphism 214, 321, 324, 331, 334, 335, 433
 POSITIVE_INFINITY 84
 private (keyword) 61, 70, 205, 211–259,
 268–301, 306–324, 343, 354–379, 385, 386,
 388–390, 400, 401, 408–448, 450, 452, 453,
 469, 470, 479
 Promotion 80, 83
 protected (keyword) 70, 241–246, 261, 269,
 278, 279, 287, 302, 318, 374, 408, 410, 413,
 435
 provides (keyword) 9, 14, 42, 70, 71, 105, 106,
 166, 296, 331, 351, 454
 public (keyword) 17, 18, 22, 23, 27, 31–33, 38,
 40–61, 64, 67–71, 98, 100, 101, 104, 108,
 113, 114, 241–251, 253, 255–258, 264–301,
 305–335, 408–448

R

Reference 97, 188, 227, 306, 307, 330, 412
 Reflection 401
 Runnable (interface) 326, 327
 Runtime 14

S

SAM (Single Abstract Method) 291, 303
 Singleton 209, 211, 256–259, 261, 262, 290, 421
 Specialization 267
 static (keyword) 18, 37, 38, 40, 61, 247–259,
 268, 276, 277, 280–283, 286–292, 296–300,
 319–321, 326, 333–336, 364, 368, 369, 373,
 374, 396, 400, 407–449, 464, 470, 473
 strictfp (keyword) 71, 86, 278
 StringTokenizer (class) 188
 super (keyword) 71, 263, 272–276, 295, 300, 302,
 303, 318, 335, 336, 343, 354, 356, 417, 461,
 472, 482
 SuppressWarnings (annotation) 364, 465

switch (keyword) 71, 121, 134, 144–150,
152–163, 169, 170, 357, 375, 377, 378, 408,
437, 438, 441, 442, 450, 464
synchronized (keyword) 71, 182, 203, 206, 278
System (class) 17, 33, 42, 103, 181, 251, 254, 309,
355, 366, 393, 394, 442, 443, 449

T

Thread 327
toString() 111, 256, 276, 323, 330, 331, 343,
345, 346, 352, 354–359, 427, 432, 435, 436,
439, 469, 470, 472, 473
transient (keyword) 71, 278
transitive (keyword) 71, 216, 464
Two's Complement 76

U

UML 121, 166–168, 170, 194, 198–202, 204, 206,
207, 217, 218, 221, 226, 230, 232, 258, 260,
265, 269, 284, 294, 338, 384
Underscore 87
Unicode 88–91, 93–96, 118
Unsigned 126

V

var 65, 66, 71, 113–117, 119, 120, 139, 140, 144,
157, 158, 242, 244–246, 279, 293, 311, 312,
322, 325, 352, 353, 355, 358, 359, 412, 425,
426, 477, 478, 486
Varargs 43, 62, 312–314
volatile (keyword) 5, 71, 74, 278

W

Warnings 363, 463
Wildcard 467, 471, 478

X

XLint 462

Y

yield 71, 152, 154–160

CLAUDIO DE SIO CESARI

JAVA FOR ALIENS

**LEARN JAVA FROM SCRATCH
AND BECOME A PRO**

Volume 2

Java for Aliens - Volume 2

Copyright © 2019 by **Claudio De Sio Cesari**

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, without the prior written permission of the author, except in the case of brief quotations permitted by copyright law. For permission requests, write to the author at the address: **claudio@claudiodesio.com**

Proofreader: Proofreading Guy (<http://www.proofreadingguy.com>)

Book and Cover Editor: Emanuele Giuliani (emanuele@giuliani.mi.it)

Cover Designers: Ciro Improta (improta@libero.it)

Riccardo Ferti (riccardo.ferti@gmail.com)

Andrea Massaretti (andrea.massaretti@gmail.com)

Printed by: kdp.amazon.com

First Edition (November, 2019)

ISBN: 979-12-200-4916-0

Font licenses

Libre Baskerville (<https://fonts.google.com/specimen/Libre+Baskerville>, Impallari Type): OFL

Libre Franklin (<https://fonts.google.com/specimen/Libre+Franklin>, Impallari Type): OFL

Cousine (<https://fonts.google.com/specimen/Cousine>, Steve Matteson): AL

Inconsolata (<https://fonts.google.com/specimen/Inconsolata>, Raph Levien): OFL

Roboto (<https://fonts.google.com/specimen/Roboto>, Christian Robertson): AL

Digits (<https://www.1001fonts.com/digits-font.html>, Dieter Steffmann): FFC

Journal Dingbats 3 (<https://www.1001fonts.com/journal-dingbats-3-font.html>, Dieter Steffmann): FFC

Image licenses

Curiosity icon (https://www.flaticon.com/free-icon/toyger-cat_107975, www.freepik.com): FBL

Alien icon (<http://www.iconarchive.com/show/free-space-icons-by-goodstuff-no-nonsense/alien-4-icon.html>, goodstuffnononsense.com): CC

Trick icon (https://www.flaticon.com/free-icon/magic-wand_1275106, www.flaticon.com/authors/pause08): FBL

Atom image ([https://commons.wikimedia.org/wiki/File:Atom_symbol_as_used_in_the_logo_of_the_television_series_The_Big_Bang_Theory_\(black\).svg](https://commons.wikimedia.org/wiki/File:Atom_symbol_as_used_in_the_logo_of_the_television_series_The_Big_Bang_Theory_(black).svg)): CCSA

Sun image (<http://www.clker.com/clipart-simple-sun-yellow.html>, SirTobi): CLKER

License specifications

Open Free License (OFL): https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL_web

Apache License, Version 2.0 (AL): <http://www.apache.org/licenses/LICENSE-2.0>

1001Fonts Free For Commercial Use License (FFC): <https://www.1001fonts.com/licenses/ffc.html>

Flaticon Basic License (FBL): <https://file000.flaticon.com/downloads/license/license.pdf>

CC Attribution 4.0 (CC): <https://creativecommons.org/licenses/by/4.0/legalcode>

CC Attribution SA 3.0U (CCSA): <https://creativecommons.org/licenses/by-sa/3.0/deed.en>

CLKER (CLKER): <http://www.clker.com/disclaimer.html>

Any other trademarks, service marks, product names or named features are assumed to be the property of their respective owners, and are used only for reference. There is no implied endorsement if we use one of these terms.

Table of Contents

Preface

XVII

Part IV - Java API Fundamentals 1

Chapter 13 - The Indispensable Library: The `java.lang` Package 3

13.1 The Object Class 4

13.1.1 The `equals()` Method 5

13.1.2 The `hashCode()` Method 6

13.1.3 The `toString()` Method 8

13.1.4 The `clone()` Method 9

13.2 The System Class 11

13.2.1 Input and Output 11

13.2.2 The Properties 12

13.2.3 Other Methods of the `System` Class 13

13.2.4 Garbage Collection 15

13.2.4.1 G1 15

13.2.4.2 Epsilon 16

13.2.4.3 Z 16

13.2.4.4 Shenandoah 17

13.3 The Runtime Class 17

13.4 The Reflection API and the `Class` Class 18

13.4.1 Creating a `Class` Object 19

13.4.2 `Class` is a Generic Type 19

13.4.3 A Reflection Example 20

13.4.4 Names of Method Parameters 22

13.5 The String Class 23

13.5.1 Memory Management 24

13.5.2 Declaration of the `String` Class 26

13.5.3 The String Methods	27
13.5.3.1 Strings and Indexes	27
13.5.3.2 Checking Strings	28
13.5.3.3 Generating a String from a String	29
13.5.3.4 Dividing Strings	32
13.5.4 Compact Strings	35
13.5.5 Text Blocks (feature preview)	37
13.5.5.1 Introduction	37
13.5.5.2 Feature Preview	38
13.5.5.3 Syntax	39
13.5.5.4 Normalization of Line Terminators	39
13.5.5.5 Removal of Incidental White Spaces	40
13.5.5.6 Interpretation of Escape Characters	43
13.5.5.7 Text Block Concatenation	43
13.5.5.8 New Methods of the String Class	44
13.6 The Wrapper Classes	46
13.6.1 Autoboxing-Autounboxing	47
13.6.1.1 Wrapper Types and Arithmetic Operations	48
13.6.1.2 Wrapper Types and Collections	49
13.6.1.3 Primitive Data Types Memory Management	50
13.6.1.4 Assignment of a null Value to the Wrapper Type	51
13.6.1.5 Constructs and Relational Operators of the Language	51
13.6.1.6 equals() Method	52
13.6.1.7 Overload	53
13.6.2 Other Methods of Wrapper Classes	53
13.6.2.1 Deprecated Constructors	53
13.6.2.2 Methods Added with Java 8	56
13.7 The Math Class	57
13.8 Sorting	58
13.8.1 The Comparable Interface	58
13.8.2 The Comparator Interface	60
Summary	62

Chapter 14 - Utilities API: java.util Package and Date-Time API	65
--	----

14.1 The java.util Package	66
14.1.1 Configuration Files	66
14.1.1.1 The Properties Class	67
14.1.1.2 The Preferences Class	69
14.1.2 Internationalization (i18n)	72
14.1.2.1 Locale Class and Internationalization	72
14.1.2.2 The ResourceBundle Class	74
14.1.3 The StringTokenizer Class	75
14.1.4 Output Formats	77

14.1.5 Regular Expressions	78
14.1.6 Dates, times and currencies	81
14.2 Date-Time API	83
14.2.1 Method Standardization	84
14.2.2 The <code>java.time</code> Package	84
14.2.2.1 The <code>Instant</code> Class	84
14.2.2.2 The <code>Duration</code> and <code>Period</code> Classes	88
14.2.2.3 The <code>DayOfWeek</code> and <code>Month</code> Enumerations	88
14.2.2.4 Date Classes: <code>LocalDate</code> , <code>YearMonth</code> , <code>MonthYear</code> and <code>Year</code>	89
14.2.2.5 The <code>LocalTime</code> and <code>LocalDateTime</code> Classes	90
14.2.2.6 Geolocation: the <code>ZonedDateTime</code> , <code>ZoneId</code> and <code>ZoneOffset</code> Classes	90
14.2.3 The <code>java.time.format</code> Package	91
14.2.4 The <code>java.time.temporal</code> Package	93
14.2.4.1 The <code>ChronoUnit</code> Enumeration	93
14.2.4.2 Temporal Adjusters	94
14.2.4.3 Temporal Queries	95
14.2.5 Legacy Code	96
Summary	97
 Chapter 15 - Thread Management	 101
15.1 Introduction to Threads	102
15.1.1 Temporary Definition of Thread	102
15.1.2 What Multi-threading Means	102
15.2 The Thread Class and the Temporal Dimension	104
15.2.1 <code>ThreadExists</code> Analysis	104
15.2.2 The <code>Runnable</code> Interface and the Creation of Threads	106
15.2.3 <code>ThreadCreation</code> Analysis	107
15.2.4 The <code>Thread</code> Class and the Creation of Threads	111
15.3 Priorities, Schedulers and Operating Systems	112
15.3.1 <code>ThreadRace</code> Analysis	114
15.3.2 Windows Behavior (and most Linux Systems)	115
15.3.3 Unix Behavior	115
15.3.4 The <code>volatile</code> Modifier	118
15.4 Thread and Synchronization	119
15.4.1 <code>Synch</code> Analysis	121
15.4.2 Monitor and Lock	126
15.5 Thread Communication	127
15.5.1 <code>IdealEconomy</code> Analysis	129

15.6 Concurrency	133
15.6.1 Immutable Objects	133
15.6.2 Java Technologies and Auxiliary Standard Classes	135
15.6.3 Introduction to Java Concurrency Library	136
15.6.3.1 The <code>java.util.concurrent.locks</code> Package	136
15.6.3.2 The <code>java.util.concurrent.atomic</code> Package	138
15.6.3.3 Executor Interfaces	138
15.6.3.4 The <code>Semaphore</code> Class	141
15.6.3.5 The <code>CyclicBarrier</code> Class	143
Summary	144
Conclusions Part IV	146

Part V - Java Language Evolution 147

Chapter 16 - Annotation Types 149

16.1 Definition of Annotation (Metadata)	150
16.1.1 First Example	153
16.1.2 Different Types of Annotations and Syntax	156
16.1.2.1 Ordinary Annotation (or Full Annotation)	156
16.1.2.2 Single-Value Annotation	159
16.1.2.3 Marker Annotation	160
16.2 Annotating Annotations (Meta-Annotations)	161
16.2.1 Target	161
16.2.1.1 Type Annotations	162
16.2.2 Retention	164
16.2.3 Documented	165
16.2.4 Inherited	166
16.2.5 Repeatable	167
16.3 Standard Annotations	170
16.3.1 Override	170
16.3.2 Deprecated	171
16.3.3 FunctionalInterface	172
16.3.4 SuppressWarnings	173
16.3.5 Native	176
Summary	176

Chapter 17 - Lambda Expressions	179
17.1 Lambda Expressions	181
17.1.1 Syntax	182
17.1.2 Understanding Lambda Expressions	183
17.1.3 Lambda Expressions v Anonymous Classes	183
17.1.3.1 Succinctness	184
17.1.3.2 Rules and Visibility	185
17.1.3.3 Dynamic Code	187
17.1.4 Exception Handling	190
17.1.5 Local Variable Type Inference for Lambda Expressions	192
17.1.6 Local Variable Syntax for Lambda Parameters	193
17.1.7 More on Parameters of Lambda Expressions	194
17.1.7.1 Mixed Solutions for Type Specification	194
17.1.7.2 Inference with Potentially Compatible Types	195
17.1.7.3 Inference with Inheritance-Related Types	196
17.1.8 When to Use Lambda Expressions	196
17.2 Method References	197
17.2.1 Syntax	198
17.2.2 Reference to a Static Method	198
17.2.3 Reference to an Instance Method of a Particular Object	199
17.2.4 Reference to an Instance Method of a Particular Type	200
17.2.5 Constructor Reference	202
17.2.6 Local Method Reference Type Inference	203
17.3 The Functional Interfaces of the <code>java.util.function</code> Package	203
17.3.1 Predicate	204
17.3.2 Consumer	205
17.3.3 Supplier	206
17.3.4 Function	207
17.3.5 <code>UnaryOperator</code> and Composition of Lambda Expressions	208
17.3.6 Functional Interfaces and Primitive Types	210
17.4 Intersection Types and Lambda Expressions	211
17.4.1 Example	211
17.4.2 The Word <code>var</code>	214
Summary	215
Chapter 18 - Collections Framework and Stream API	219
18.1 Introduction to the Collections Framework	220
18.2 The Collection Interface	222
18.2.1 Dependency Inversion Principle	223
18.2.2 Collection Methods	223

18.2.3 Iterating Collections: the Iterable Interface	226
18.2.3.1 Foreach Loop (Enhanced for Loop)	227
18.2.3.2 Iterators	227
18.2.3.3 The Enumeration Interface	229
18.2.3.4 The forEachRemaining() Method of the Interface Iterator Interface	230
18.2.3.5 The forEach() Method of the Iterable Interface	230
18.2.3.6 Implementation of an Iterable Type	231
18.3 List Interface	232
18.3.1 List Implementations	232
18.3.1.1 The ArrayList Class	232
18.3.1.2 The LinkedList Class	234
18.3.1.3 The Vector Class	235
18.4 The Set and SortedSet Interfaces	236
18.4.1 Set and SortedSet Implementations	236
18.4.1.1 The HashSet Class	236
18.4.1.2 The TreeSet Class	237
18.4.1.3 The LinkedHashSet Class	240
18.5 The Queue and Deque Interfaces	240
18.5.1 The Queue and Deque Implementations	242
18.5.1.1 The PriorityQueue Class	242
18.5.2 The BlockingQueue Interface	243
18.5.2.1 BlockingQueue Implementations	244
18.5.3 The Deque Implementations	244
18.6 The Map and SortedMap Interfaces	245
18.6.1 Map and SortedMap Implementations	247
18.6.1.1 The Hashtable Class	247
18.6.1.2 The HashMap Class	248
18.6.1.3 The TreeMap Class	248
18.6.1.4 The ConcurrentHashMap Class	249
18.7 Algorithms and Utilities	250
18.7.1 Collections	250
18.7.1.1 Wrapper Methods	250
18.7.1.2 Synchronization Methods	252
18.7.1.3 Convenience Methods	254
18.7.2 Arrays	254
18.7.3 Convenience Methods to Create Immutable Collections	255
18.7.3.1 Set Interface	255
18.7.3.2 Map Interface	256
18.7.3.3 List Interface	257
18.7.3.4 Custom Collections	258

18.8 Introduction to the Stream API	258
18.8.1 Stream and Pipeline Definitions	260
18.8.2 Optional Classes	261
18.8.2.1 Other <code>or()</code> Methods	263
18.8.2.2 The <code>get()</code> , <code>filter()</code> , <code>map()</code> and <code>flatMap()</code> Methods	264
18.8.3 Reduction Operations	266
18.8.3.1 <code>reduce()</code> Method	266
18.8.3.2 The <code>collect()</code> Method and the <code>Collectors</code> Class	267
18.8.4 Parallel Streams and Performance	268
Summary	270
 Chapter 19 - Java Platform Module System	 275
19.1 Why a Modular System?	276
19.1.1 Pros	276
19.1.2 Cons	278
19.2 Module Definition	278
19.2.1 <code>HelloModularWorld</code>	279
19.2.2 Architectural and Design Concepts	282
19.2.2.1 Vertical Partitioning and Subsystems	282
19.2.2.2 Dependency, Cohesion and Coupling	282
19.2.2.3 Law of Demeter	283
19.2.2.4 Horizontal Partitioning and Frameworks	284
19.2.2.5 JAR and Software Components	284
19.2.3 Directives	285
19.2.3.1 <code>requires</code>	285
19.2.3.2 <code>exports</code>	287
19.2.3.3 <code>opens</code> and <code>open</code> (Strong Encapsulation)	288
19.2.3.4 <code>uses</code> and <code>provides to</code>	290
19.3 Services with <code>ServiceLoader</code>	290
19.3.1 Factory Pattern	291
19.3.2 Modular Factory	291
19.3.3 Evolution with <code>ServiceLoader</code>	295
19.3.4 Compilation and Execution	298
19.3.5 <code>provider()</code> Method	298
19.4 Modular JAR	299
19.4.1 <code>jar</code> Tool	300
19.4.2 <code>jmod</code> Format	301
19.5 Migration of Pre-Java 9 Programs	301
19.5.1 The <code>jdeps</code> tool	302
19.5.2 Unnamed Module	303
19.5.3 Automatic Module	303

19.5.4 Testing External Modules Quickly	303
19.5.5 Modules and <code>var</code>	304
19.6 Jlink Tool	305
Summary	306
Part V Conclusions	308

Part VI - Java Integration API 309

Chapter 20 - Input-Output 311

20.1 Introduction to Input-Output	311
20.2 Decorator Pattern	312
20.2.1 Pattern Description	312
20.2.2 Example	315
20.3 Features of the <code>java.io</code> Package	316
20.3.1 Character Streams	318
20.3.2 Byte Streams	319
20.3.3 The Interfaces at the Base of the Hierarchy	321
20.3.4 Checked Exception and Closing of Streams	322
20.4 Classic Examples of Inputs and Outputs	325
20.4.1 Reading Keyboard Input	325
20.4.1.1 <code>BufferedReader</code> and <code>InputStreamReader</code>	325
20.4.1.2 The <code>Scanner</code> Class	327
20.4.1.3 The <code>Console</code> Object	327
20.4.2 File Management	328
20.4.3 Object Serialization and <code>transient</code> Modifier	332
20.4.3.1 The <code>Serializable</code> Interface	333
20.4.3.2 The <code>Transient</code> Modifier	333
20.4.3.3 Serialization Example	334
20.4.3.4 Deserialization Example	334
20.4.3.5 The <code>SerialVersionUID</code> Field	335
20.4.3.6 Customized Serialization	336
20.4.3.7 The <code>Externalizable</code> Interface	337
20.4.4 Deep Copy	340
20.4.5 Base 64 Encoding	341
20.5 New Input - Output	343
20.5.1 The <code>java.nio</code> Package	343
20.5.2 NIO 2	344
20.5.2.1 The <code>Path</code> Interface	344
20.5.2.2 The <code>Files</code> Class	347

20.6 Introduction to Networking	352
20.6.1 Basics: Client and Server	352
20.6.2 Network Protocols	353
20.6.3 Knowing the IP Address	354
20.6.4 Example (Socket API)	354
20.6.4.1 Server	355
20.6.4.2 Client	356
20.6.5 Conclusions	357
20.7 Introduction to Java Native Interface (JNI)	358
20.7.1 Example	359
20.7.1.1 Creation of File Header	359
20.7.1.2 Compilation and Creation of Header Files	360
20.7.1.3 Native Code as a Library	361
20.7.1.4 Execution	361
Summary	362
 Chapter 21 - Java Database Connectivity	 365
21.1 Introduction to JDBC	366
21.2 JDBC Basics	367
21.2.1 Vendor Implementation (JDBC Driver)	367
21.2.1.1 Types of JDBC Drivers	368
21.2.2 Developer Implementation (JDBC Application)	369
21.2.3 Analysis of the JDBCApp Example	370
21.2.4 Database Schema	373
21.3 Independence from the Database	373
21.4 Support for SQL	374
21.4.1 DML Operations	375
21.4.2 DDL Operations	376
21.4.3 Parameterized Statements	377
21.4.4 Stored Procedures	378
21.4.5 Java – SQL Types Mapping	379
21.5 Transactions with JDBC	381
21.5.1 Preserving Data with Transaction Isolation Levels	385
21.5.2 Savepoints	388
21.6 JDBC Evolution	390
21.6.1 JDBC 2.0	391
21.6.2 JDBC 3.0	392
21.6.2.1 Updatable ResultSet	392
21.6.2.2 The Insert Row	393
21.6.3 JDBC 4.0	394
21.6.4 JDBC 4.1	395

21.6.5 JDBC 4.2	396
21.6.6 JDBC 4.3	397
Summary	397

Chapter 22 - Java and the XML World 399

22.1 JAXP: java.xml Module	400
22.1.1 Document Object Model (DOM)	401
22.1.1.1 Creating a DOM Document from an XML File	401
22.1.1.2 Retrieving the List of Nodes from a DOM Document	402
22.1.1.3 Retrieving Selected Nodes	403
22.1.1.4 Editing an XML Document	405
22.1.2 Simple API for XML (SAX)	407
22.1.3 Streaming API for XML (StAX)	409
22.1.4 XSLT Transformations	414
22.1.4.1 XPath	415
22.1.4.2 Writing a Document Type File with XSLT	417
22.1.4.3 Printing on the Screen with XSLT	418
22.1.4.4 Transform a file using XSLT	419
22.2 JAXB: java.xml.bind Module	421
22.2.1 Module Management	422
22.2.2 Data Classes: Song and Playlist	423
22.2.3 Main Class: JAXBExample	425
22.2.4 Compilation and Execution	426
Summary	428
Part VI Conclusions	429

Part VII - Java GUIs 431

Chapter 23 - Graphical User Interfaces (GUIs) 433

23.1 Introduction to Graphical User Interfaces	433
23.2 Introduction to the Abstract Window Toolkit (AWT)	436
23.2.1 AWT Examples	437
23.2.1.1 Graphical Components	438
23.2.1.2 Useful AWT Types Even Without the AWT-Based GUIs	439
23.2.1.3 Draw with Canvas	440
23.2.1.4 Composite Pattern	440

23.3 Complex Interfaces with Layout Managers	441
23.3.1 FlowLayout	443
23.3.2 BorderLayout	445
23.3.3 GridLayout	446
23.3.4 Creation of Complex Graphic Interfaces	447
23.3.5 GridBagLayout	449
23.3.6 CardLayout	449
23.4 Events Handling	451
23.4.1 Nested Classes: Introduction and History	451
23.4.2 Observer and Listener	453
23.4.3 Nested Classes and Anonymous Classes	456
23.4.4 Lambda Expressions	459
23.4.5 Other Event Types	460
23.4.6 Adapter Classes	462
23.5 Introduction to Swing	463
23.5.1 Swing vs AWT	464
23.5.2 Top-Level Containers	467
23.5.3 First Example: SwingExample	469
23.5.4 Concurrency of Swing Components	471
23.5.5 Pluggable Look and Feel: LookAndFeelExample	472
23.5.6 Main Swing Graphical Components: SwingMixExample	475
23.5.6.1 LabelsPanel : JLabel and ImageIcon Classes	476
23.5.6.2 ButtonsPanel : JButton Class and Shortcuts	477
23.5.6.3 PaintDialog : JDialog , JList and Color Classes	479
23.5.6.4 TextPanel : JTextField , JTextArea and Border	482
23.5.6.5 BoxesPanel : JComboBox , JRadioButton and JCheckBox Classes	483
23.5.6.6 TreePanel : JTree Class	484
23.5.6.7 TablePanel : JTable Class	486
23.6 Other Features of the GUIs	487
23.6.1 Splash Screen	487
23.6.2 Background Applications: SystemTrayExample	487
23.6.3 Launching Applications with a Graphical User Interface	488
23.6.3.1 Startup Script	489
23.6.3.2 Executable JAR File	489
Summary	490
 Chapter 24 - Introduction to JavaFX	 493
<hr/>	
24.1 History of Java GUIs	494
24.2 JavaFX Features	496

24.3 First JavaFX Program	497
24.3.1 FirstJFXExample Analysis	498
24.3.2 Life cycle of a JavaFX Application	498
24.3.2.1 The <code>main()</code> Method and Command-Line Arguments	500
24.3.3 Running a JavaFX Application	502
24.3.3.1 Running Non-Modular JavaFX Applications from the Command Line	502
24.3.3.2 Create an Executable Non-Modular JAR File	503
24.3.3.3 Create an Executable Module	504
24.3.3.4 Create an Executable Modular JAR	505
24.3.3.5 Create a Custom Runtime Image	506
24.4 Complex Interfaces Using Layouts	507
24.4.1 The JavaFX Layout Pane	507
24.4.1.1 The <code>VBox</code> and <code>HBox</code> Classes	508
24.4.1.2 The <code>BorderPane</code> Class	510
24.4.1.3 The <code>GridPane</code> Class	511
24.4.2 FXML	513
24.5 CSS	517
24.5.1 CSS and FXML	517
24.5.2 CSS and JavaFX	518
24.6 Event Handling	519
24.7 JavaFX Properties and Bindings	520
24.7.1 JavaFX Properties	520
24.7.2 Binding	522
24.8 Special Effects	523
24.8.1 Blur	523
24.8.2 Fading	523
24.8.3 Translation	524
24.9 Advanced Graphical Components	524
24.9.1 Pie Chart	524
24.9.2 Media Player	525
24.9.3 Browser	526
Summary	527
Part VII Conclusions	528
 Java Version New Feature Index	 531

Word Index	533
-------------------	-----

Java Version New Feature Index

List of pages where new features are present for any Java version.



4, 15, 53, 171, 172, 180, 255, 275



16, 192, 203, 214, 223, 304, 360



16, 29, 31, 33, 34, 193, 345, 421, 496, 497, 526



16, 17, 45



17, 37, 44, 46, 355

Word Index

Symbols

-> (used for switch and lambda) 182–215, 230, 238, 261–266, 269, 326, 352, 459, 470–480, 488, 520
-Xlint 6, 171, 172, 174, 335
== (equal to) 5, 7, 24, 25, 51, 52
@interface (keyword) 153, 155–157, 159–166, 168–171, 173, 175, 176
\n 13, 14, 33, 34, 37, 40, 43, 45, 46, 63, 75, 326
\r 13, 14, 33, 34, 40, 43, 75
\t 39, 43–45, 75, 79, 338
^ (XOR) 7, 55

A

abstract (keyword) 4, 5, 18, 58, 91, 96, 105, 135, 141, 151–157, 173, 182–215, 220–223, 227, 235, 258, 290, 291, 295, 313, 314, 318, 359, 440, 441, 462, 478, 479, 498, 499, 520, 522
algorithm 7, 15–17, 21, 40, 43, 46, 63, 95, 141, 196, 215, 238, 254, 255, 268, 272, 273, 318, 320, 326, 341
Annotation 152, 156, 159–161, 167
args[] 20, 330, 357
Array 59
arrayCopy() 14, 62
Arrays (class) 4, 14, 59–62, 66, 141, 199, 219, 226, 242, 243, 250, 254, 255, 257, 273
Attribute 517
Autounboxing 47

B

BlockingQueue (interface) 136, 242–244, 271
boolean (keyword) 5, 22, 23, 28–30, 32, 34, 39, 50, 51, 53, 76, 78, 87, 113, 118, 119, 131, 133, 137, 154, 171, 188, 189, 196, 197, 204, 210, 211, 216, 224, 225, 228, 229, 231, 236, 240–243, 246, 249, 259, 264, 265, 329, 331, 332, 379, 380, 386
break (keyword) 109, 292, 325, 404, 412, 416

C

Class (class) 4, 6, 8, 11, 13, 17–26, 44, 57, 63–69, 72–75, 84, 104, 111, 141, 143, 154, 169, 181, 184, 212, 228, 232–237, 240, 242, 247–249, 267, 288, 289, 300, 327, 347, 360, 370, 371, 374, 394, 395, 397, 425, 440, 477, 484–490, 501–505, 510–513
ClassCastException 239
clone() 9–11, 14, 62, 135, 340
cohesion 112, 283, 528
Collection (interface) 15, 221–228, 230–236, 240, 241, 246, 250, 252, 253, 258, 260, 270
Collections (class) 4, 6, 14, 49, 56, 60–62, 66, 67, 101, 136, 147, 200, 201, 226, 229, 232, 235, 240, 242, 250–258, 270, 271, 273, 284, 308, 528
Compact Strings 35, 63
Comparable (interface) 3, 27, 28, 35, 55, 56, 58–60, 63, 64, 66, 199, 222, 238, 239, 242
Comparator (interface) 3, 4, 60–64, 66, 199, 222, 234, 237–239, 242, 246, 271

Constructor 18, 20, 22, 202, 230, 499, 500
Coupling 282, 307
currentTimeMillis() 14, 15, 36, 62, 233
CyclicBarrier (class) 143–145

D

Date-Time API 83, 84, 93, 96, 98–100, 133, 255, 272
Decorator (pattern) 311–316, 318, 362, 364
deep copy 10, 340, 341, 363
Deprecated (annotation) 53, 170–172, 177, 279
Deque (interface) 221, 222, 235, 240–242, 244, 245, 270, 271
DIP (Dependency Inversion Principle) 223, 228, 276, 283, 291, 307
Documented (annotation) 161, 162, 165, 168, 169, 171–173, 176, 177
Duration (class) 88, 523, 524

E

Eclipse 7, 336, 357, 464, 503
EJE 38, 67–69, 73–75, 80, 98, 135, 136, 325, 327, 344, 351, 357, 404, 408, 437, 439, 465, 476, 487, 489, 503
Enhanced for Loop 227
enum (keyword) 153, 157, 159–161, 164, 168, 479
Enumeration 93, 220, 229, 247, 270
equals() 5–7, 23, 25, 26, 29, 30, 40, 52, 62, 224, 236, 242, 259
Executors 138, 139, 141, 191, 358
exports (keyword) 279, 287, 296, 299, 302, 303, 306, 307, 504
extends (keyword) 5, 9, 141, 220, 222, 226, 227, 232, 236–249, 340, 403, 406, 407, 476, 498, 511, 516

F

Factory Method (pattern) 202, 291
Foreach 227
Fork/Join 141

G

Garbage Collector 15
Generics 161
Geolocation 90, 99

H

HashMap (class) 6, 62, 66, 67, 97, 154, 220, 228, 236, 247–250, 256, 271
HashSet (class) 222, 224, 230, 236, 237, 239, 240, 249, 255, 258, 271
Hashtable (class) 67, 97, 101, 220, 229, 240, 247, 248, 250, 271
HelloModularWorld 279–281, 288, 304, 306
HelloWorld 359–361, 498, 499, 501, 502

I

i18n 72, 74, 97
immutable 24, 47, 50, 52, 63, 83–85, 133, 134, 145, 250–252, 254–257, 271, 272
immutable copies 250, 252
implements (keyword) 16, 27, 31, 95, 106, 137, 156, 211, 221, 231, 240, 241, 244, 259, 294, 297, 338, 367, 403, 435, 441, 454, 457, 461–463, 471, 477, 478, 521
import (keyword) 162, 289, 293, 422, 476, 504, 515
inheritance 196, 284, 307, 312, 313, 316, 362, 528
instanceof (keyword) 5, 163, 164, 404
Instant (class) 84, 87, 88, 93, 336, 337, 352
Internationalization 72, 97, 434
Iterator 76, 80, 174, 175, 224, 227–232, 239, 253, 254, 269, 270, 327

J

JAR 172, 275–277, 284, 285, 299–303, 307, 308, 367, 368, 489–491, 503, 505
Jlink 305
JMOD 147, 275, 276, 301, 307, 308
JShell 54, 55, 60, 256, 528

L

Lambda expression 184
 List (interface) 50, 61, 163, 164, 173–175, 196,
 200, 219, 221–223, 225, 232, 234, 245,
 250–255, 257, 258, 267, 270, 271, 402, 411,
 412, 501, 531
 LocalDateTime (class) 90–99, 396
 Locale (class) 72–75, 77, 82, 92, 93, 97
 LocalTime (class) 90, 93–98, 208, 209, 396
 Lock 126, 136, 137, 141, 145

M

Map (interface) 67, 154, 155, 219, 221, 222,
 245–250, 252, 254–258, 267, 270, 271
 Math (class) 1, 3, 57, 58, 63, 64, 77, 133, 259
 Metadata 150, 176
 Method 5, 6, 8–10, 18, 21, 22, 52, 63, 84, 154, 155,
 161, 183, 197–200, 202, 203, 230, 266, 267,
 288, 291, 298, 360, 380, 381, 500
 module (keyword) 4, 35, 65, 147, 239, 275,
 277–282, 285–287, 289–296, 298–308, 367,
 368, 400, 416, 421–423, 426–429, 433, 491,
 502–507, 516, 520, 523, 527
 module descriptor 278, 280, 285, 286, 293, 296,
 306, 504
 Monitor 126

N

Native (annotation) 10, 161, 170, 176, 177, 358,
 361, 364, 369
 native (keyword) 176, 177, 276, 301, 307, 309,
 311, 358, 359, 364, 369, 429, 451, 464, 465,
 494, 528
 NavigableMap (interface) 248
 Netbeans 7, 336, 422, 426, 435, 513, 521
 NIO 2 309, 311, 343, 344, 363, 364
 notify() 130–133, 135–137, 141, 145
 notifyAll() 130, 135, 137, 145

O

Object (class) 3–11, 14, 19–23, 29, 32, 47, 62, 64,
 77, 103, 130, 145, 150, 151, 163, 174, 180,
 199, 203, 212, 213, 224, 226, 240, 246, 249,
 255, 327, 332, 335, 337, 340, 342, 347, 363,
 380, 400, 401, 428, 434, 451, 452, 486, 528
 OCL (Object Constraint Language) 151
 Overload 53
 Override 5, 9, 137, 139, 143, 144, 149, 150, 160,
 170, 171, 177, 181, 184, 191, 200, 202,
 297–299, 411, 424, 440, 458, 459, 463, 472,
 473, 478, 480, 483, 497, 499, 501, 510

P

Package 84, 91, 93, 136, 138, 161, 203, 279, 316,
 343, 390
 ParallelGC 15, 16, 489
 Path (interface) 344–352, 363, 490
 Period (class) 88, 98
 Pipeline 260
 private (keyword) 11, 22, 57, 112, 134, 156, 173,
 183, 288–290, 336, 456, 458, 474, 520
 protected (keyword) 9, 62, 156, 289, 344
 provides (keyword) 278, 279, 284, 290, 295, 296,
 299
 public (keyword) 9, 17, 156, 157, 181, 184, 232,
 288, 289, 303, 366

Q

Queue (interface) 136, 221, 222, 235, 240–244,
 270, 271

R

Recursive 141, 402, 403, 535
 Reference 198–200, 202, 203
 Reflection 1, 18, 20, 63, 155, 169, 308
 Repeatable (annotation) 161, 167, 168, 177
 ResourceBundle (class) 73–75, 97, 98
 Retention (annotation) 155, 161, 162, 164–166,
 168–172, 175–177

Runnable (interface) 4, 101, 103, 106, 108, 111–113, 120, 127, 128, 135–141, 143, 144, 146, 181–183, 190, 191, 193, 200, 203, 470–472
Runtime 1, 3, 16–18, 36, 63, 64, 506

S

SAM (Single Abstract Method) 140, 173, 183, 184, 196, 197, 203, 215, 481, 484, 520
Semaphore (class) 141, 142, 145
ServiceLoader 147, 290, 291, 295–298, 307
Singleton 17
SortedMap (interface) 221, 222, 245–250, 253, 270, 271
SortedSet (interface) 221, 222, 224, 236, 239, 246, 249, 250, 253, 270, 271
static (keyword) 8, 11, 14, 19, 29, 39, 54, 56, 57, 59, 67, 72, 73, 96, 109, 111, 157, 162, 173, 179, 183, 198–209, 250, 255, 292, 298, 327, 330, 335, 342, 359, 371, 387, 418, 501, 513
Stream API 66, 147, 222, 227, 258, 270, 272, 273, 528
StringTokenizer (class) 65, 75, 76, 79, 80, 98, 100, 327
Strong encapsulation 277
super (keyword) 9, 200, 224, 246, 264, 265, 362, 367, 480, 509, 511
SuppressWarnings (annotation) 160, 173–175
switch (keyword) 38, 51, 292, 412, 473, 474, 528
Synchronization 119, 145, 252
synchronized (keyword) 101, 103, 119, 120, 124–127, 129–132, 135–137, 139, 145, 146, 233, 235, 247, 248, 253, 254, 270, 350
System (class) 11–15, 77, 203, 237, 327, 426, 499

T

Target (annotation) 161–166, 168–173, 175–177
Temporal Adjusters 94
TemporalQueries (class) 95, 96, 99
Thread 4, 111–114, 119, 120, 127, 128, 130, 132, 135, 140, 142–146, 181–183, 186, 187, 190–192, 200, 333, 334, 363, 450, 500
toString() 8, 9, 32, 61, 62, 105, 150, 188, 205, 208, 226, 230, 239, 252, 268, 299, 334, 338, 345, 346, 352, 411, 424, 484
transient (keyword) 332–335, 343, 363
transitive (keyword) 278, 286, 293
TreeMap (class) 239, 248, 249, 271
TreeSet (class) 236–240, 249, 267, 271

U

UML 151, 220, 282, 284, 314, 528
Unicode 27, 34, 39, 60, 237, 238, 318, 496
URI 345, 347

V

var (keyword) 25, 192–196, 203, 214–216, 223, 255, 304, 305, 339, 473, 474
volatile (keyword) 101, 113, 114, 118, 119, 145, 146

W

wait() 115, 130–133, 135–137, 141, 145