



Exploring Java 9

Build Modularized Applications in Java

—
Fu Cheng

Apress®

Exploring Java 9

Build Modularized Applications in Java



Fu Cheng

Apress®

Exploring Java 9

Fu Cheng
Auckland, New Zealand

ISBN-13 (pbk): 978-1-4842-3329-0 ISBN-13 (electronic): 978-1-4842-3330-6
<https://doi.org/10.1007/978-1-4842-3330-6>

Library of Congress Control Number: 2017962328

Copyright © 2018 by Fu Cheng

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.

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Technical Reviewer: Massimo Nardone
Coordinating Editor: Jessica Vakili
Copy Editor: Rebecca Rider
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. 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 rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

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/978-1-4842-3329-0. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

To my wife Andrea and my daughters Olivia and Erica

Contents

About the Author	xiii
About the Technical Reviewer	xv
■ Chapter 1: Introduction	1
Installation.....	1
IDE	2
IntelliJ IDEA	2
Eclipse	2
Build Tools	4
Gradle	4
Apache Maven	4
javac and java.....	4
Docker	5
CI Builds	6
Summary.....	6
■ Chapter 2: The Module System.....	7
Module Introduction	8
Sample Application.....	8
Module Declaration	9
requires and exports.....	9
Transitive Dependencies.....	9
Static Dependencies.....	12
Services.....	12

Qualified Exports	14
Opening Modules and Packages	14
Working with Existing Code.....	15
Unnamed Modules.....	15
Automatic Modules.....	16
JDK Tools.....	17
Module Paths.....	17
Module Version	18
The Main Module.....	18
Root Modules.....	18
Limiting the Observable Modules	19
Upgrading the Module Path	19
Increasing Readability and Breaking Encapsulation.....	19
javac	20
jlink.....	21
java	24
jdeps.....	24
Module Java API	27
ModuleFinder.....	27
ModuleDescriptor	28
Configuration	31
The Module Layers	34
Class Loaders	39
Class.....	42
Reflection	43
Automatic Module Names	43
Module Artifacts	46
JAR Files.....	46
JMOD Files.....	47
JDK Modules.....	49

Common Issues	49
Migration in Action	51
Building the Project Using Java 9	51
The Migration Path	51
BioJava	52
Summary	56
■ Chapter 3: jshell	57
Code Completion	58
Classes	58
Methods	59
Commands	59
/list	59
/edit	60
/drop	61
/save	61
/open	61
/imports	62
/vars	62
/types	62
/methods	63
/history	63
/env	63
/set	64
/reset	64
/reload	64
/!	65
/<id>	65
/-<n>	65
/exit	65
Summary	65

■ **Chapter 4: Collections, Stream, and Optional** **67**

 Factory Methods for Collections..... 67

 The List.of() Method..... 67

 The Set.of() Method 67

 The Map.of() and Map.ofEntries() Methods..... 68

Arrays **68**

 Mismatch() Methods..... 68

 Compare() Methods 69

 Equals() Methods..... 69

Stream..... **69**

 The ofNullable() Method 69

 The dropWhile() Method 70

 The takeWhile() Method..... 70

 The iterate() Method 71

 IntStream, LongStream, and DoubleStream 71

Collectors **71**

 The filtering() Method 71

 The flatMapping() Method 72

Optional **73**

 The ifPresentOrElse() Method..... 73

 The Optional.or() Method 73

 The stream() Method 74

Summary..... **74**

■ **Chapter 5: The Process API** **75**

 The ProcessHandle Interface 75

 Process..... 77

 Managing Long-Running Processes..... 78

Summary..... **79**

■ Chapter 6: The Platform Logging API and Service	81
Default LoggerFinder Implementation.....	82
Creating Custom LoggerFinder Implementations.....	83
Summary.....	86
■ Chapter 7: Reactive Streams	87
Core Interfaces.....	87
Flow.Publisher<T>	87
Flow.Subscriber<T>.....	87
Flow.Subscription	88
Flow.Processor<T,R>	88
SubmissionPublisher.....	88
Third-Party Libraries	95
RxJava 2	95
Reactor	96
Interoperability	97
Summary.....	97
■ Chapter 8: Variable Handles	99
Creating Variable Handles	99
findStaticVarHandle	99
findVarHandle	99
unreflectVarHandle	100
Access Modes	100
Memory Ordering.....	100
VarHandle Methods	101
Arrays	105
byte[] and ByteBuffer Views	106
Memory Fence	107
Summary.....	107

■ Chapter 9: Enhanced Method Handles	109
arrayConstructor	109
arrayLength	109
varHandleInvoker and varHandleExactInvoker	110
zero	110
empty	111
Loops.....	111
loop.....	111
countedLoop	113
iteratedLoop.....	114
whileLoop and doWhileLoop.....	115
Try-finally	116
Summary.....	117
■ Chapter 10: Concurrency	119
CompletableFuture	119
Async	119
Timeout.....	119
Utilities.....	120
TimeUnit and ChronoUnit.....	120
Queues	121
Atomic Classes	122
Thread.onSpinWait	123
Summary.....	124
■ Chapter 11: Nashorn	125
Getting the Nashorn Engine	125
ECMAScript 6 Features.....	126
Template Strings.....	126
Binary and Octal Literals	126
Iterators and for.of Loops.....	126
Functions	127

Parser API.....	128
Basic Parsing.....	128
Parsing Error.....	129
Analyzing Function Complexity	130
Summary.....	131
■ Chapter 12: I/O	133
InputStream.....	133
The ObjectInputStream Filter	134
Summary.....	137
■ Chapter 13: Security.....	139
SHA-3 Hash Algorithms	139
SecureRandom.....	139
Using PKCS12 as the Default Keystore.....	141
Summary.....	141
■ Chapter 14: User Interface	143
Desktop	143
Application Events.....	143
About Window.....	144
Preferences Window.....	144
Open Files.....	145
Print Files.....	146
Open URI.....	146
Application Exit.....	146
Other Functionalities	148
Multiresolution Images.....	148
TIFF Image Format	150
Deprecating the Applet API.....	150
Summary.....	150

■ Chapter 15: JVM	151
Unified Logging	151
Tags, Levels, Decorations, and Output.....	151
Logging Configuration	152
The Diagnostic Command VM.log.....	153
Remove GC Combinations	154
Making G1 the Default Garbage Collector	154
Deprecating the Concurrent Mark Sweep (CMS) Garbage Collector	154
Removing Launch-Time JRE Version Selection	154
More Diagnostic Commands	155
Removal of the JVM TI hprof Agent	157
Removal of the jhat Tool	157
Removal of Demos and Samples	157
Javadoc	157
Summary.....	159
■ Chapter 16: Miscellaneous	161
Small Language Changes.....	161
Private Interface Methods	161
Resource References in try-with-resources.....	161
Other Changes	162
The Stack-Walking API	162
Objects	164
Unicode 8.0	165
UTF-8 Property Resource Bundles	166
Enhanced Deprecation	166
NetworkInterface.....	167
Summary.....	168
Index	169

About the Author

Fu Cheng is a full-stack software developer working in a healthcare start-up in Auckland, New Zealand. During his many years of experiences, he has worked in different companies to build large-scale enterprise systems, government projects, and SaaS products. He is an experienced JavaScript and Java developer and always wants to learn new things. He enjoys sharing knowledge by writing blog posts, technical articles, and books.

About the Technical Reviewer



Massimo Nardone has more than 22 years of experiences in Security, Web/Mobile development, and Cloud and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching others how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

Massimo holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has held the positions of Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Massimo's 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 currently works as the Chief Information Security Office (CISO) for Cargotec Oyj.

He has also been 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).

Massimo has reviewed more than 40 IT books for different publishing company and he is the coauthor of *Pro Android Games* (Apress, 2015).

He dedicates this book to Antti Jalonen and his family who are always there when Massimo needs them.

CHAPTER 1



Introduction

Java SE 9 was released on September 21, 2017. It's the first major release of the Java platform since Java SE 8 was released on March 18, 2014. The Java community has been waiting for this release for quite a long time. The release of Java 9 was delayed several times. In this release, the Java Platform Module System (JPMS), or Project Jigsaw, introduces a module system to the Java platform. Now we can create modular Java applications using Java 9. The contents of Java SE 9 can be found in JSR 379: Java™ SE 9 Release Contents (<https://www.jcp.org/en/jsr/detail?id=379>). When I'm talking about Java 9 in this book, I'm referring to Java SE 9 and JDK 9.

Installation

You can download the JDK 9 General Availability (GA) release builds from Oracle (<http://www.oracle.com/technetwork/java/javase/downloads/jdk9-downloads-3848520.html>). You can choose the build for your platform and install it on your local machine.

After the installation, you can run `java -version` to check the version. Listing 1-1 shows the version information of JDK 9 on macOS.

Listing 1-1. JDK 9 Version on macOS

```
java version "9.0.1"
Java(TM) SE Runtime Environment (build 9.0.1+11)
Java HotSpot(TM) 64-Bit Server VM (build 9.0.1+11, mixed mode)
```

■ **Note** Java 9 uses a new schema for version strings. The version strings before Java 9 start with 1, for example, 1.8.0_60 for Java 8. In Java 9, the leading 1 of the original version strings has been removed. The version strings for Java 9 start with 9. Java 9 also provides the class `Runtime.Version` to represent and parse the version strings.

Installing Java 9 GA builds makes them the default JVM on your local machine, which may break some Java applications that are not yet compatible with Java 9. You can use a tool like jEnv (<http://www.jenv.be/>) to manage multiple JDK installations or update the environment variable `JAVA_HOME` manually to point to your old JDK installations. Some of these applications may have options for configuring the JRE for use. For these applications, you can use the options to point to the old JDK installations.

IDE

IDEs are very useful for Java developers. I'll discuss Java 9 support of two popular IDEs: IntelliJ IDEA and Eclipse.

IntelliJ IDEA

IntelliJ IDEA (<https://www.jetbrains.com/idea/>) 2017.1 already supports Java 9. You can simply install Java 9 GA builds and add JDK 9 as the project SDK. Don't forget to change the language level to 9; see Figure 1-1.

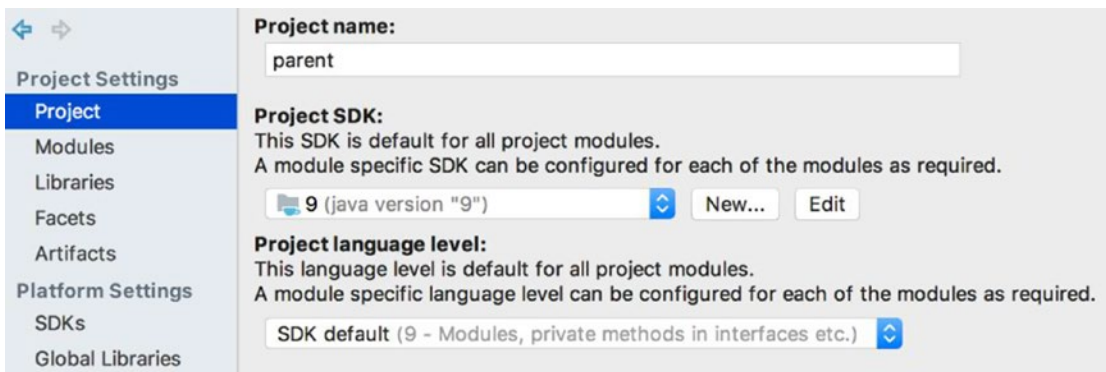


Figure 1-1. IntelliJ IDEA Java 9 setup

All the code in this book is written and tested using IDEA 2017.2.

Eclipse

At the time of writing, when compared to using IntelliJ IDEA, using Eclipse for Java 9 development requires more manual setup steps. To run Java 9 you need at least Eclipse Oxygen (4.7) with Eclipse Marketplace Client installed. When you're ready to get started, you first need to modify the file `eclipse.ini` to specify the JVM and add extra arguments. If Java 9 is not the default JVM, you need to use the option `-vm` to specify the JDK 9 installation path, for example, `-vm /Library/Java/JavaVirtualMachines/jdk-9.0.1.jdk/Contents/Home/bin/javaw`. You also need to add the JVM argument `--add-modules=ALL-SYSTEM`, which should be placed after `-vmargs`. After you've done this, you can start Eclipse using JDK 9. Finally, you need to open **Eclipse Marketplace** in the **Help** menu, search for "Java 9," and then install the **Java 9 Support (BETA) for Oxygen 4.7**; see Figure 1-2. After you restarting Eclipse, you can add a new Java project with Java SE 9 as the execution environment and the compiler compliance level set to 9.

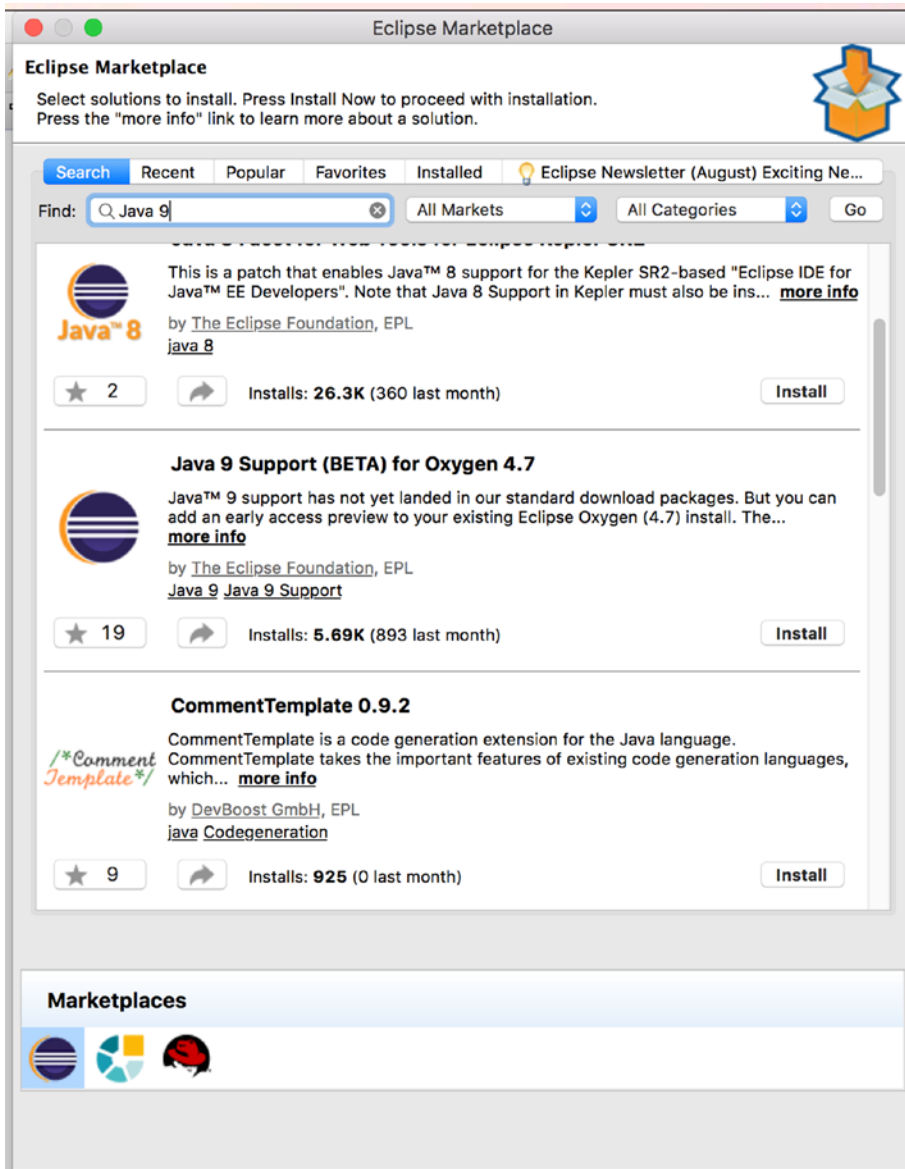


Figure 1-2. Install Eclipse support for Java 9

Eclipse continues to improve its support for Java 9. You should always refer to the latest official guide for its Java 9 support.

Build Tools

Typically, you need to use some kinds of build tools when you're developing nontrivial Java applications. The popular choices are Gradle (<https://gradle.org/>) and Apache Maven (<https://maven.apache.org/>).

Gradle

The support of Gradle for Java 9 is tracked in this GitHub issue (<https://github.com/gradle/gradle/issues/719>). According to the latest status (<https://github.com/gradle/gradle/issues/719#issuecomment-312225355>), you should use at least Gradle version 4.1-milestone-1 to run with Java 9.

For the most part, version 4.1-milestone-1 requires no special environment settings to get Gradle running on a simple Java project. However, there are still many, commonly-used plugins for which at least --permit-illegal-access will be required to run.

The option --permit-illegal-access has been renamed to --illegal-access and already has the default value permit, so it's no longer required to add this option. At the time of writing, Gradle 4.2 doesn't provide first-class support of modules. You can experiment with Gradle's module support using this guide (<https://guides.gradle.org/building-java-9-modules/>).

Apache Maven

The Maven Compiler plugin (<https://maven.apache.org/plugins/maven-compiler-plugin/>) supports compiling Java 9 modules. Listing 1-2 shows how to configure this plugin in Maven projects.

Listing 1-2. Maven Compiler Plugin Configuration

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.7.0</version>
  <configuration>
    <source>9</source>
    <target>9</target>
    <fork>true</fork>
  </configuration>
</plugin>
```

javac and java

If you just want to run some quick tests to see how Java 9 works, you can skip the setup with Gradle or Maven and use javac and java directly. Java code can be compiled using javac and run using java. javac and java commands both have new options, which I cover in later chapters.

You should only use javac and java for small projects. For large projects, these command line arguments are very hard to manage. Use Gradle or Maven for nontrivial projects.