



Learn Java for Android Development

Migrating Java SE Programming Skills
to Mobile Development

—
Fourth Edition
—

Peter Späth
Jeff Friesen

Apress®

Learn Java for Android Development

**Migrating Java SE Programming
Skills to Mobile Development**

Fourth Edition

**Peter Späth
Jeff Friesen**

Apress®

Learn Java for Android Development: Migrating Java SE Programming Skills to Mobile Development

Peter Späth
Leipzig, Sachsen, Germany

Jeff Friesen
Winnipeg, MB, Canada

ISBN-13 (pbk): 978-1-4842-5942-9
<https://doi.org/10.1007/978-1-4842-5943-6>

ISBN-13 (electronic): 978-1-4842-5943-6

Copyright © 2020 by Peter Späth and Jeff Friesen

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
Editorial Operations Manager: Mark Powers

Cover designed by eStudioCalamar

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 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/9781484259429. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

- About the Authors.....xv
- About the Technical Reviewerxvii
- Introductionxix
- Fourth Edition Notesxxi
- Chapter 1: Getting Started with Java 1
 - What Is Java?..... 2
 - Java Is a Language..... 2
 - Java Is a Platform..... 3
 - Java SE and Java EE..... 6
 - Installing the JDK and Exploring Example Applications 7
 - Hello, World! 8
 - DumpArgs..... 12
 - EchoText 13
 - Installing and Exploring the Eclipse IDE..... 16
 - Java Meets Android..... 21
 - What Is Android? 21
 - History of Android 22
 - Android Architecture..... 22
 - Android Says Hello 26
 - Summary..... 28

TABLE OF CONTENTS

Chapter 2: Learning Language Fundamentals 31

 Learning Application Structure 31

 Learning Comments 33

 Single-Line Comments 33

 Multiline Comments 34

 Javadoc Comments 34

 Learning Identifiers 38

 Learning Types 39

 Primitive Types 40

 Object Types 42

 Array Types 43

 Learning Variables 43

 Learning Expressions 44

 Simple Expressions 45

 Compound Expressions 51

 Learning Statements 80

 Assignment Statements 80

 Decision Statements 81

 Loop Statements 87

 Break Statements 94

 Continue Statements 96

 Summary 98

Chapter 3: Discovering Classes and Objects 101

 Declaring Classes 102

 Classes and Applications 102

 Constructing Objects 103

 Default Constructor 105

 Explicit Constructors 105

 Objects and Applications 109

Encapsulating State and Behaviors	111
Representing State via Fields.....	111
Representing Behaviors via Methods	121
Hiding Information	136
Initializing Classes and Objects	142
Class Initializers	143
Instance Initializers	144
Initialization Order	146
Collecting Garbage.....	146
Revisiting Arrays	147
Summary.....	154
Chapter 4: Discovering Inheritance, Polymorphism, and Interfaces	157
Building Class Hierarchies	157
Extending Classes	158
The Ultimate Superclass.....	165
Composition.....	173
Changing Form.....	174
Upcasting and Late Binding.....	176
Abstract Classes and Abstract Methods.....	180
Downcasting and Runtime Type Identification	182
Covariant Return Types.....	185
Formalizing Class Interfaces.....	186
Declaring Interfaces	186
Implementing Interfaces.....	188
Extending Interfaces.....	192
Why Use Interfaces?	193
Summary.....	201

Chapter 5: Mastering Advanced Language Features, Part 1..... 203

Mastering Nested Types..... 203

 Static Member Classes..... 203

 Nonstatic Member Classes..... 208

 Anonymous Classes..... 210

 Local Classes..... 212

 Interfaces Within Classes 214

Mastering Packages 215

 What Are Packages?..... 215

 Package Names Must Be Unique 216

 The Package Statement 216

 The Import Statement..... 217

 Searching for Packages and Types..... 218

 Playing with Packages 220

 Packages and JAR Files 225

Mastering Static Imports..... 226

Mastering Exceptions..... 227

 What Are Exceptions?..... 228

 Representing Exceptions in Source Code..... 228

 Throwing Exceptions 233

 Handling Exceptions 236

 Performing Cleanup..... 241

 Automatic Resource Management 244

Summary..... 247

Chapter 6: Mastering Advanced Language Features, Part 2..... 249

Mastering Annotations 249

 Discovering Annotations..... 249

 Declaring Annotation Types and Annotating Source Code 253

 Processing Annotations 259

Mastering Generics	261
Collections and the Need for Type Safety	261
Generic Types	263
Generic Methods	275
Arrays and Generics	279
Mastering Enums	282
The Trouble with Traditional Enumerated Types	282
The Enum Alternative	284
The Enum Class	290
Summary.....	296
Chapter 7: Exploring the Basic APIs, Part 1.....	299
Exploring Math	299
Exploring Number and Its Children	302
BigDecimal	303
BigInteger	307
Primitive Type Wrapper Classes	311
Exploring String, StringBuffer, and StringBuilder	319
String	319
StringBuffer and StringBuilder	322
Exploring System	324
Exploring Threads	327
Runnable and Thread.....	328
Synchronization	334
Thread-Local Variables	350
Summary.....	355
Chapter 8: Exploring the Basic APIs, Part 2.....	359
Exploring Random	359
Exploring Reflection	363
The Class Entry Point.....	363
Constructor, Field, and Method.....	369

TABLE OF CONTENTS

Package	373
Array	378
Exploring StringTokenizer	379
Exploring Timer and TimerTask	381
Summary.....	384
Chapter 9: Exploring the Collections Framework	387
Exploring Collections Framework Fundamentals.....	387
Comparable vs. Comparator	389
Iterable and Collection.....	391
Exploring Lists.....	395
ArrayList	396
LinkedList	397
Exploring Sets	398
TreeSet	398
HashSet	400
EnumSet	400
Exploring Sorted Sets.....	401
Exploring Navigable Sets	404
Exploring Queues	406
PriorityQueue.....	407
Exploring Deques	409
ArrayDeque	410
Exploring Maps	411
TreeMap.....	414
HashMap	414
IdentityHashMap.....	418
WeakHashMap.....	419
EnumMap	419
Exploring Sorted Maps	419
Exploring Navigable Maps	420

Exploring the Arrays and Collections Utility APIs.....	420
Exploring the Legacy Collection APIs	423
Summary.....	428
Chapter 10: Functional Programming	431
Functions and Operators.....	432
Lambda Calculus.....	432
Entering a Stream	433
Mapping	435
Filtering.....	436
Terminating a Stream.....	437
Performing Actions on Each Element.....	437
Limiting, Skipping, Sorting, and Distinct	438
Ranges	438
Reducing	438
Collecting	439
Methods As Functions.....	442
Single-Method Interfaces.....	443
Streams and Parallelization	444
Protonpack, a Stream Utility Library	444
Summary.....	446
Chapter 11: Exploring the Concurrency Utilities.....	449
Introducing the Concurrency Utilities.....	449
Exploring Executors	451
Exploring Synchronizers	454
Countdown Latches.....	454
Cyclic Barriers	454
Exchangers	455
Semaphores	455

TABLE OF CONTENTS

Exploring the Concurrent Collections.....	456
Exploring the Locking Framework.....	457
Lock.....	457
ReentrantLock	459
Condition	459
ReadWriteLock	460
ReentrantReadWriteLock.....	460
Exploring Atomic Variables.....	461
Summary.....	464
Chapter 12: Performing Classic I/O	467
Working with the File API	467
Constructing File Instances	468
Learning About Stored Abstract Pathnames.....	470
Learning About a Pathname's File or Directory	471
Obtaining Disk Space Information.....	472
Listing Directories	473
Creating and Manipulating Files and Directories.....	475
Setting and Getting Permissions	476
Working with the RandomAccessFile API.....	478
Working with Streams.....	479
Stream Classes Overview.....	479
ByteArrayOutputStream and ByteArrayInputStream.....	481
FileOutputStream and FileInputStream	483
PipedOutputStream and PipedInputStream.....	485
FilterOutputStream and FilterInputStream	488
BufferedOutputStream and BufferedInputStream	491
DataOutputStream and DataInputStream	492
Object Serialization and Deserialization.....	495
PrintStream	501
Standard I/O Revisited.....	503

Working with Writers and Readers	505
Writer and Reader Classes Overview	506
Writer and Reader	508
OutputStreamWriter and InputStreamReader.....	508
FileWriter and FileReader	510
Summary.....	516
Chapter 13: Accessing Networks	519
Accessing Networks via Sockets	519
Socket Addresses	521
Socket Options	522
Socket and ServerSocket	524
DatagramSocket and MulticastSocket	530
Accessing Networks via URLs.....	535
URL and URLConnection	535
URLEncoder and URLDecoder.....	539
URI	541
Accessing Network Interfaces and Interface Addresses.....	542
Managing Cookies.....	546
Summary.....	551
Chapter 14: Migrating to New I/O	555
Working with Buffers	556
Buffer and Its Children.....	557
Working with Channels	559
Channel and Its Children	560
Working with Selectors.....	566
Selector Fundamentals.....	566
Selector Demonstration.....	572
Working with Regular Expressions	577
Pattern, PatternSyntaxException, and Matcher	577
Character Classes.....	580

TABLE OF CONTENTS

- Capturing Groups..... 582
- Boundary Matchers and Zero-Length Matches 583
- Quantifiers 584
- Practical Regular Expressions 587
- Working with Charsets..... 588
 - A Brief Review of the Fundamentals 588
 - Working with Charsets 589
 - Charsets and the String Class 593
- Working with Formatter and Scanner..... 595
 - Working with Formatter..... 596
 - Working with Scanner 601
- Summary..... 605
- Chapter 15: Accessing Databases 607**
 - Introducing Apache Derby..... 608
 - Apache Derby Installation and Configuration 611
 - Apache Derby Demos 611
 - Apache Derby Command-Line Tools..... 611
 - Starting an Apache Derby Server 613
 - Embedded Apache Derby Example..... 614
 - Introducing SQLite 615
 - Accessing Databases via JDBC..... 617
 - Data Sources, Drivers, and Connections..... 617
 - Statements 621
 - Metadata 632
 - Summary..... 637
- Chapter 16: Working with XML and JSON Documents..... 641**
 - What Is XML? 641
 - XML Declaration 643
 - Elements and Attributes 645
 - Character References and CDATA Sections..... 647

Namespaces	649
Comment and Processing Instructions	654
Well-Formed Documents	655
Valid Documents	656
Parsing XML Documents with SAX	658
Exploring the SAX API	658
Demonstrating the SAX API	661
Parsing and Creating XML Documents with DOM	672
A Tree of Nodes	673
Exploring the DOM API	676
What Is JSON?	688
JSON Processing in Java	689
Generating JSON	689
Parsing JSON	693
Summary	697
Chapter 17: Date and Time	699
The Traditional Date and Time API	699
About Dates and Calendars	700
Date and Time Formatters	708
Parsing	710
The New Date and Time API	713
Local Dates and Times	713
Instants	716
Offset Dates and Times	717
Zoned Dates and Times	720
Duration and Periods	722
Clock	725
Summary	728

TABLE OF CONTENTS

Appendix A: Solutions to Exercises 729

Chapter 1: Getting Started with Java 729

Chapter 2: Learning Language Fundamentals 731

Chapter 3: Discovering Classes and Objects..... 734

Chapter 4: Discovering Inheritance, Polymorphism, and Interfaces..... 741

Chapter 5: Mastering Advanced Language Features, Part 1 750

Chapter 6: Mastering Advanced Language Features, Part 2 758

Chapter 7: Exploring the Basic APIs, Part 1 765

Chapter 8: Exploring the Basic APIs, Part 2..... 775

Chapter 9: Exploring the Collections Framework..... 780

Chapter 10: Functional Programming 788

Chapter 11: Exploring the Concurrency Utilities..... 791

Chapter 12: Performing Classic I/O 794

Chapter 13: Accessing Networks 805

Chapter 14: Migrating to New I/O..... 811

Chapter 15: Accessing Databases..... 818

Chapter 16: Working with XML and JSON Documents 820

Chapter 17: Date and Time..... 826

Index..... 831

About the Authors

Peter Späth consults, trains/teaches, and writes books on various subjects, with a primary focus on software development. With a wealth of experience in Java-related languages, authoring the new edition of a “Java for Android” book seemed to be a very good idea with respect to improving the community’s and other audience’s proficiency for developing Android apps. He also graduated in 2002 as a physicist and soon afterward became an IT consultant, mainly for Java-related projects.

Jeff Friesen is a freelance tutor and software developer with an emphasis on Java (and now Android). In addition to authoring *Learn Java for Android Development* and co-authoring *Android Recipes*, Jeff has written numerous articles on Java and other technologies for JavaWorld, informIT, Java.net, and DevSource.

About the Technical Reviewer

Chád (“Shod”) Darby is an author, instructor, and speaker in the Java development world. As a recognized authority on Java applications and architectures, he has presented technical sessions at software development conferences worldwide (in the United States, the United Kingdom, India, Russia, and Australia). In his 15 years as a professional software architect, he’s had the opportunity to work for Blue Cross/Blue Shield, Merck, Boeing, Red Hat, and a handful of startup companies.

Chád is a contributing author to several Java books, including *Professional Java E-Commerce* (Wrox Press), *Beginning Java Networking* (Wrox Press), and *XML and Web Services Unleashed* (Sams Publishing). He has Java certifications from Sun Microsystems and IBM. Chád holds a BS in computer science from Carnegie Mellon University. You can visit his blog at www.luv2code.com to view his free video tutorials on Java. You can also follow him on Twitter at @darbyluvs2code.

Introduction

Smartphones and tablets are all the rage these days. Their popularity is largely due to their ability to run apps.

Android app developers are making money by selling such apps or intra-app features.

In today's challenging economic climate, you might like to try your hand at developing Android apps and generate some income. If you have good ideas, perseverance, and some artistic talent (or perhaps know some talented individuals), you are already part of the way toward achieving this goal.

Most importantly, you'll need to possess a solid understanding of the Java language and foundational application programming interfaces (APIs) before jumping into Android. After all, many Android apps are written in Java and interact with many of the standard Java APIs (such as threading and input/output APIs).

We wrote *Learn Java for Android Development* to give you a solid Java foundation that you can later extend with knowledge of Android architecture, API, and tool specifics. This book will give you a strong grasp of the Java language and the many important APIs that are fundamental to Android apps and other Java applications. It will also introduce you to key development tools.

Fourth Edition Notes

With current Android versions, the new features Java 8 introduced also entered the Android development world. For this reason, Java 8 features were added in the new book edition, namely, functional programming aspects, the streaming API, the new date and time API, and JSON handling. To streamline the book, more or less verbatim copies of parts of the official documentation were replaced by small URL references. Besides, corner cases, while valuable for certain development scenarios, were removed—after all the book is targeting Java beginners and all too advanced topics might confuse more than help to learn Java fundamentals.

Having the official Java documentation open in a browser window while reading the book certainly is a good idea.

Android meanwhile has become the most important smartphone development platform, so if you want to learn smartphone app development, starting with a decent Android version and for the very popular Java platform is a very good idea.

We hope the new book release will help you to readily acquire enough Java language programming skill, so you can explore the fun of Android application development soon.

Book Organization

The first edition of this book was organized into 10 chapters and 1 appendix. The second edition was organized into 14 chapters and 3 appendixes. The third edition was organized into 16 chapters and 2 appendixes with a bonus appendix on Android app development. For the fourth edition, a new Chapter 10 is added for functional programming and streams, and Chapter 17 contains a new introduction to the new date and time API (as of Java 8). Important topics of the former Chapter 16 were moved at appropriate places in the other chapters. Each chapter in each edition offers a set of exercises that you should complete to get the most benefit from its content. Their solutions are presented in Appendix A.

Chapter 1 introduces you to Java by first focusing on Java's dual nature (language and platform). It then briefly introduces you to Oracle's Java SE and Java EE editions of the Java platform. You next learn how to download and install the Java SE Development Kit (JDK), and you learn some Java basics by developing and playing with three simple Java applications. After receiving a brief introduction to the Eclipse IDE, you receive a brief introduction to Android.

Chapter 2 starts you on an in-depth journey of the Java language by focusing on language fundamentals. You first learn about simple application structure and then learn about comments, identifiers (and reserved words), types, variables, expressions (and literals), and statements.

Chapter 3 continues your journey by focusing on classes and objects. You learn how to declare a class and organize applications around multiple classes. You then learn how to construct objects from classes, declare fields in classes and access these fields, declare methods in classes and call them, initialize classes and objects, and remove objects when they're no longer needed. You also learn more about arrays, which were first introduced in Chapter 2.

Chapter 4 adds to Chapter 3's pool of object-based knowledge by introducing you to the language features that take you from object-based applications to object-oriented applications. Specifically, you learn about features related to inheritance, polymorphism, and interfaces. While exploring inheritance, you learn about Java's ultimate superclass. Also, while exploring interfaces, you discover why they were included in the Java language; interfaces are not merely a workaround for Java's lack of support for multiple implementation inheritance, but serve a higher purpose.

Chapter 5 introduces you to four categories of advanced language features: nested types, packages, static imports, and exceptions.

Chapter 6 introduces you to three additional advanced language feature categories: annotations, generics, and enums.

Chapter 7 begins a trend that focuses more on APIs than language features. This chapter first introduces you to Java's Math-oriented types. It then explores Number and its various subtypes (such as Integer, Double, and BigDecimal). Next you explore the string-oriented types (String, StringBuffer, and StringBuilder) followed by the System type. Finally, you explore the Thread class and related types for creating multithreaded applications.

Chapter 8 continues to explore Java's basic APIs by focusing on the `Random` class for generating random numbers; the `References` API, `Reflection`, and the `StringTokenizer` class for breaking a string into smaller components; and the `Timer` and `TimerTask` classes for occasionally or repeatedly executing tasks.

Chapter 9 focuses exclusively on Java's Collections Framework, which provides you with a solution for organizing objects in lists, sets, queues, and maps. You also learn about collection-oriented utility classes and review Java's legacy collection types.

Chapter 10 introduces the functional programming features which entered the Java world with Java 8. We also introduce the streaming API, which closely connects to functional programming.

Chapter 11 focuses exclusively on Java's Concurrency Utilities. After receiving an introduction to this framework, you explore executors, synchronizers (such as countdown latches), concurrent collections, the locking framework, and atomic variables (where you discover compare-and-swap).

Chapter 12 is all about classic input/output (I/O), largely from a file perspective. In this chapter, you explore classic I/O in terms of the `File` class, `RandomAccessFile` class, various stream classes, and various writer/reader classes. Our discussion of stream I/O includes coverage of Java's object serialization and deserialization mechanisms.

Chapter 13 continues to explore classic I/O by focusing on networks. You learn about the `Socket`, `ServerSocket`, `DatagramSocket`, and `MulticastSocket` classes along with related types. You also learn about the `URL` class for achieving networked I/O at a higher level and learn about the related `URI` class. After learning about the low-level `NetworkInterface` and `InterfaceAddress` classes, you explore cookie management, in terms of the `CookieHandler` and `CookieManager` classes, and the `CookiePolicy` and `CookieStore` interfaces.

Chapter 14 introduces you to new I/O. You learn about buffers, channels, selectors, regular expressions, charsets, and the `Formatter` and `Scanner` types in this chapter.

Chapter 15 focuses on databases. You first learn about the Apache Derby and SQLite database products, and then explore JDBC for communicating with databases created via these products.

Chapter 16 emphasizes Java's support for XML and JSON. We first provide a tutorial on the XML topic where you learn about the XML declaration, elements and attributes, character references and CDATA sections, namespaces, comments and processing instructions, well-formed documents, and valid documents (in terms of document

type definition and XML Schema). We then show you how to parse XML documents via the SAX API, parse and create XML documents via the DOM API, use the XPath API to concisely select nodes via location path expressions, and transform XML documents via XSLT. We then talk about JSON generation and parsing.

Chapter 17 completes the chapter portion of this book by covering Java's date and time APIs. We talk about the old API, and the new API which was introduced with Java 8.

Appendix A presents solutions to all of the exercises in Chapters 1 through 17.

Note You can download this book's source code by pointing your web browser to www.apress.com/us/book/9781484259429 and clicking the **Download Source Code** button.

What Comes Next?

After you complete this book, we recommend that you check out Apress's other Android-oriented books, such as *Beginning Android* by Grant Allen, and learn more about developing Android apps.

Thanks for purchasing this fourth edition of *Learn Java for Android Development*. We hope you find it a helpful preparation for, and we wish you lots of success in achieving, a satisfying and lucrative career as an Android app developer.

Jeff Friesen and Peter Späth, May 2020

CHAPTER 1

Getting Started with Java

Android apps are written in Java and use various Java application program interfaces (APIs). Because you'll want to write your own apps, but may be unfamiliar with the Java language and these APIs, this book teaches you about Java as a first step into Android app development. It provides you with Java language fundamentals and Java APIs that are useful when developing apps.

Note This book illustrates Java concepts via non-Android Java applications. It's easier for beginners to grasp these applications than corresponding Android apps. However, we also reveal a trivial Android app toward the end of this chapter for comparison purposes.

An *API* is an interface that application code uses to communicate with other code, which is typically stored in a software library. For more information on this term, check out Wikipedia's "Application programming interface" topic at http://en.wikipedia.org/wiki/Application_programming_interface.

This chapter sets the stage for teaching you the essential Java concepts that you need to understand before embarking on an Android app development career. We first answer the question: "What is Java?" Next, we show you how to install the Java SE Development Kit (JDK) and introduce you to JDK tools for compiling and running Java applications.

After presenting a few simple example applications, we show you how to install and use the open source Eclipse IDE (integrated development environment) so that you can more easily (and more quickly) develop Java applications and (eventually) Android apps. We then provide you with a brief introduction to Android and show you how Java fits into the Android development paradigm.

What Is Java?

Java is a language and a platform originated by Sun Microsystems and later acquired by Oracle. In this section, we briefly describe this language and reveal what it means for Java to be a platform. To meet various needs, Oracle organized Java into two main editions: Java SE and Java EE. A third edition, Java ME for embedded devices, plays no prominent role nowadays. This section briefly explores each of these two editions.

Java Is a Language

Java is a language in which developers express *source code* (program text). Java's *syntax* (rules for combining symbols into language features) is partly patterned after the C and C++ languages in order to shorten the learning curve for C/C++ developers.

The following list identifies a few similarities between Java and C/C++:

- Java and C/C++ share the same single-line and multiline comment styles. Comments let you document source code.
- Many of Java's reserved words are identical to their C/C++ counterparts (for, if, switch, and while are examples) and C++ counterparts (catch, class, public, and try are examples).
- Java supports character, double-precision floating-point, floating-point, integer, long integer, and short integer primitive types via the same char, double, float, int, long, and short reserved words.
- Java supports many of the same operators, including arithmetic (+, -, *, /, and %) and conditional (?:) operators.
- Java uses brace characters ({ and }) to delimit blocks of statements.

The following list identifies a few of the differences between Java and C/C++:

- Java supports an additional comment style known as Javadoc.
- Java provides reserved words not found in C/C++ (extends, strictfp, synchronized, and transient are examples).

- Java doesn't require machine-specific knowledge. It supports the byte integer type (see [http://en.wikipedia.org/wiki/Integer_\(computer_science\)](http://en.wikipedia.org/wiki/Integer_(computer_science))), doesn't provide a signed version of the character type, and doesn't provide unsigned versions of integer, long integer, and short integer. Furthermore, all of Java's primitive types have guaranteed implementation sizes, which is an important part of achieving portability (discussed later). The same cannot be said of equivalent primitive types in C and C++.
- Java provides operators not found in C/C++. These operators include `instanceof` and `>>>` (unsigned right shift).

You'll learn about single-line, multiline, and Javadoc comments in Chapter 2. Also, you'll learn about reserved words, primitive types, operators, blocks, and statements in that chapter.

Java was designed to be a safer language than C/C++. It achieves safety in part by not letting you overload operators and by omitting C/C++ features such as *pointers* (storage locations containing addresses; see [http://en.wikipedia.org/wiki/Pointer_\(computer_programming\)](http://en.wikipedia.org/wiki/Pointer_(computer_programming))).

Java also achieves safety by modifying certain C/C++ features. For example, loops must be controlled by Boolean expressions instead of integer expressions where 0 is false and a nonzero value is true. (There is a discussion of loops and expressions in Chapter 2.)

These and other fundamental language features support classes, objects, inheritance, polymorphism, and interfaces. Java also provides advanced features related to nested types, packages, static imports, exceptions, assertions, annotations, generics, enums, and more. Subsequent chapters explore most of these language features.

Java Is a Platform

Java is a platform consisting of a virtual machine and an execution environment. The *virtual machine* is a software-based processor that presents an instruction set, and it is commonly referred to as the *Java virtual machine (JVM)*. The *execution environment* consists of libraries for running programs and interacting with the underlying operating system (also known as the *native platform*).

The execution environment includes a huge library of prebuilt class files that perform common tasks, such as math operations (e.g., trigonometry) and network communications. This library is commonly referred to as the *standard class library*.

A special Java program known as the *Java compiler* translates source code into *object code* consisting of instructions that are executed by the JVM and associated data. These instructions are known as *bytecode*. Figure 1-1 shows this translation process.

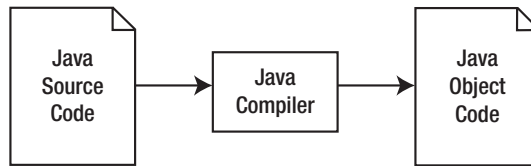


Figure 1-1. The Java compiler translates Java source code into Java object code consisting of bytecode and associated data

The compiler stores a program’s bytecode and data in files having the `.class` extension. These files are known as *class files* because they typically store the compiled equivalent of classes, a language feature discussed in Chapter 3.

A Java program executes via a tool that loads and starts the JVM and passes the program’s main class file to the machine. The JVM uses its *classloader* component to load the class file into memory.

After the class file has been loaded, the JVM’s *bytecode verifier* component makes sure that the class file’s bytecode is valid and doesn’t compromise security. The verifier terminates the JVM when it finds a problem with the bytecode.

Assuming that all is well with the class file’s bytecode, the JVM’s *interpreter* component interprets the bytecode one instruction at a time. *Interpretation* consists of identifying bytecode instructions and executing equivalent native instructions.

Note *Native instructions* (also known as *native code*) are the instructions understood by the native platform’s physical processor.

When the interpreter learns that a sequence of bytecode instructions is executed repeatedly, it informs the JVM’s *just-in-time (JIT) compiler* to compile these instructions into native code.

JIT compilation is performed only once for a given sequence of bytecode instructions. Because the native instructions execute instead of the associated bytecode instruction sequence, the program executes much faster.

During execution, the interpreter might encounter a request to execute another class file's bytecode. When that happens, it asks the classloader to load the class file and the bytecode verifier to verify the bytecode before executing that bytecode.

Also during execution, bytecode instructions might request that the JVM open a file, display something on the screen, or perform another task that requires cooperation with the native platform. The JVM responds by transferring the request to the platform via its *Java Native Interface (JNI)* bridge to the native platform. Figure 1-2 shows these JVM tasks.

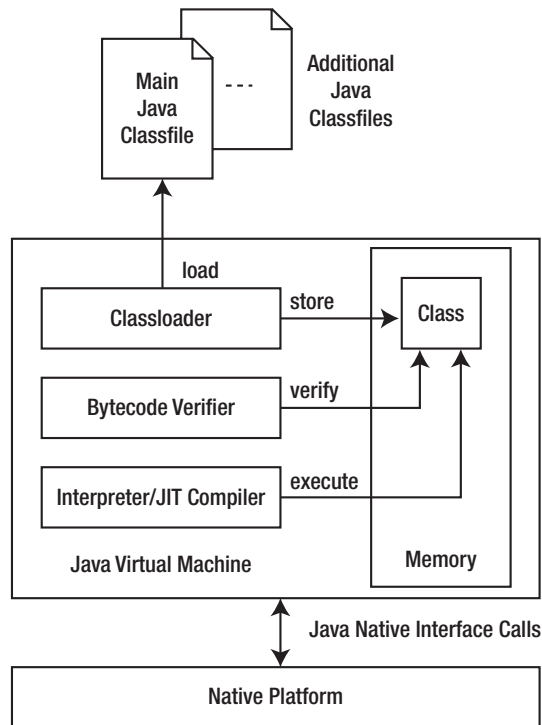


Figure 1-2. *The JVM provides all of the necessary components for loading, verifying, and executing a class file*

The platform side of Java promotes *portability* by providing an abstraction over the underlying platform. As a result, the same bytecode runs unchanged on Windows, Linux, Mac OS X, and other platforms.

Note Java was introduced with the slogan “write once, run anywhere.” Although Java goes to great lengths to enforce portability (such as defining an integer always to be 32 binary digits [bits] and a long integer always to be 64 bits (see <http://en.wikipedia.org/wiki/Bit> to learn about binary digits), it doesn’t always succeed. For example, despite being mostly platform independent, certain parts of Java (such as the scheduling of threads, discussed in Chapter 7) vary from underlying platform to underlying platform.

The platform side of Java also promotes *security* by doing its best to provide a secure environment (such as the bytecode verifier) in which code executes. The goal is to prevent malicious code from corrupting the underlying platform (and possibly stealing sensitive information).

Note Many security issues that have plagued Java have prompted Oracle to release various security updates.

Java SE and Java EE

Developers use different editions of the Java platform to create Java programs that run on desktop computers and web servers.

- *Java Platform, Standard Edition (Java SE)*: The Java platform for developing *applications*, which are stand-alone programs that run on desktops.
- *Java Platform, Enterprise Edition (Java EE)*: The Java platform for developing enterprise-oriented applications and *servlets*, which are server programs that conform to Java EE’s Servlet API. Java EE is built on top of Java SE.

This book largely focuses on Java SE and applications.

Note The open source variant of Java SE gets called OpenJDK; the open source variant of Java EE has the name Jakarta EE.

Installing the JDK and Exploring Example Applications

The Java SE edition can be downloaded from Oracle at www.oracle.com/technetwork/java/javase/overview/index.html. It contains everything needed to compile and run Java programs. For the corresponding open source variant, go to <https://openjdk.java.net/install/>.

Click the appropriate Download button to download the current JDK's installer application for your platform. Then run this application to install the JDK.

The installation directory contains various files, depending on the version you have chosen. For example, for JSE 13 you will find, among others, the following two important subdirectories:

- **bin:** This subdirectory contains assorted JDK tools. You'll use only a few of these tools in this book, mainly `javac` (Java compiler) and `java` (Java application launcher). However, you'll also work with `jar` (Java ARchive [JAR] creator, updater, and extractor [a *JAR file* is a ZIP file with special features]), `javadoc` (Java documentation generator), and `serialver` (serial version inspector).
- **lib:** This subdirectory contains Java and native platform library files that are used by JDK tools. Here you can also find the sources.

Note `javac` is not actually the Java compiler. It's a tool that loads and starts the JVM, identifies the compiler's main class file to the JVM, and passes the name of the source file being compiled to the compiler's main class file.

You can execute JDK tools at the *command line*, passing *command-line arguments* to a tool. For a quick refresher on the command line and command-line arguments topics, check out Wikipedia's "Command-line interface" entry (http://en.wikipedia.org/wiki/Command-line_interface).

The following command line shows you how to use `javac` to compile a source file named `App.java`:

```
javac App.java
```

The `.java` file extension is mandatory. The compiler complains when you omit this extension.

Tip You can compile multiple source files by specifying an asterisk in place of the file name, as follows:

```
javac *.java
```

Assuming success, an `App.class` file is created. If this file describes an application, which minimally consists of a single class containing a method named `main`, you can run the application as follows:

```
java App
```

You must not specify the `.class` file extension. The `java` tool complains when `.class` is specified.

In addition to downloading and installing the JDK, you'll need to access the JDK documentation, especially to explore the Java APIs. There are two sets of documentation that you can explore.

- Oracle's JDK documentation at <https://docs.oracle.com/javase/>
- Google's Java Android API documentation at <https://developer.android.com/reference/packages.html>

Oracle's JDK documentation presents many APIs that are not supported by Android. Furthermore, it doesn't cover APIs that are specific to Android. This book focuses only on core Oracle Java APIs that are also covered in Google's documentation.

Hello, World!

It's customary to start exploring a new language and its tools by writing, compiling, and running a simple application that outputs the "Hello, World!" message. This practice dates back to Brian Kernighan's and Dennis Ritchie's seminal book, *The C Programming Language*.

Listing 1-1 presents the source code to a HelloWorld application that outputs this message.