CHAPTER 1

Introduction to Java

"Experience is simply the name we give our mistakes."
—Oscar Wilde

1.1 What Is Java?

Java is a high-level, object-oriented, general-purpose programming language that was originally developed by James Gosling, a Canadian computer scientist, at what was then Sun Microsystems, in the U.S. state of California in 1991. Sun Microsystems was later acquired by Oracle Corporation, also in California, in 2010. Java was a byproduct of Sun's "Green" project, and it was originally designed as a platform-independent language for programming household electronic appliances. However, Java was too advanced for such applications. Gosling designed Java syntax based on the C and C++ languages, but with fewer low-level facilities. Java was named after the popular Indonesian Java coffee.

Java first appeared in 1995, through the HotJava and Netscape web browsers, as a plug-in called Java Applets, which could add dynamic content and interactions to static, pale web pages. Java soon became popular with all the major web browsers incorporating the ability to run Java applets. You've probably seen the famous Java logo, a cup of hot coffee, along with the Java mascot, Duke. Today, after decades of effect, Java has been developed into a fully functional, multipurpose, and powerful language suitable for both individual and enterprise users. Java is different from JavaScript, which is a script language that runs only within a web browser.

The Java language has five main principles; it was designed to be all of the following:

- Simple, object-oriented, and familiar
- Robust and secure
- Architecture-neutral and portable
- High-performance
- Interpreted, threaded, and dynamic

The main advantage of Java is its platform independence; that is, programs written in the language can be "write once, run anywhere" (WORA). This independence is achieved through the concept of the Java Virtual Machine (JVM), illustrated in Figure 1.1. With conventional programming languages like C/C++, to run on different operating systems such as Windows, Mac, and Linux, the C/C++ source file needs to be compiled separately on each operating system. Because each executable file runs in its native operating system, the executable files compiled in one operating system cannot run in another operating system. Java works differently. The Java source code (a `.java` file) is compiled into Java *bytecode* (a `.class` file). The bytecode files are not executable files and cannot run directly in the operating system. Instead, they run in the JVM, which handles the differences between operating systems and presents an identical environment for Java programs to run in. JVM is a novel idea that makes Java platform-independent. The drawback of the JVM is that Java programs run much more slowly than the corresponding C programs; but for most applications, this difference is not noticeable.

Java is one of the most popular programming languages, especially for networking applications. According to Oracle, worldwide there are an estimated 9 million Java developers and about 3 billion devices that run Java.

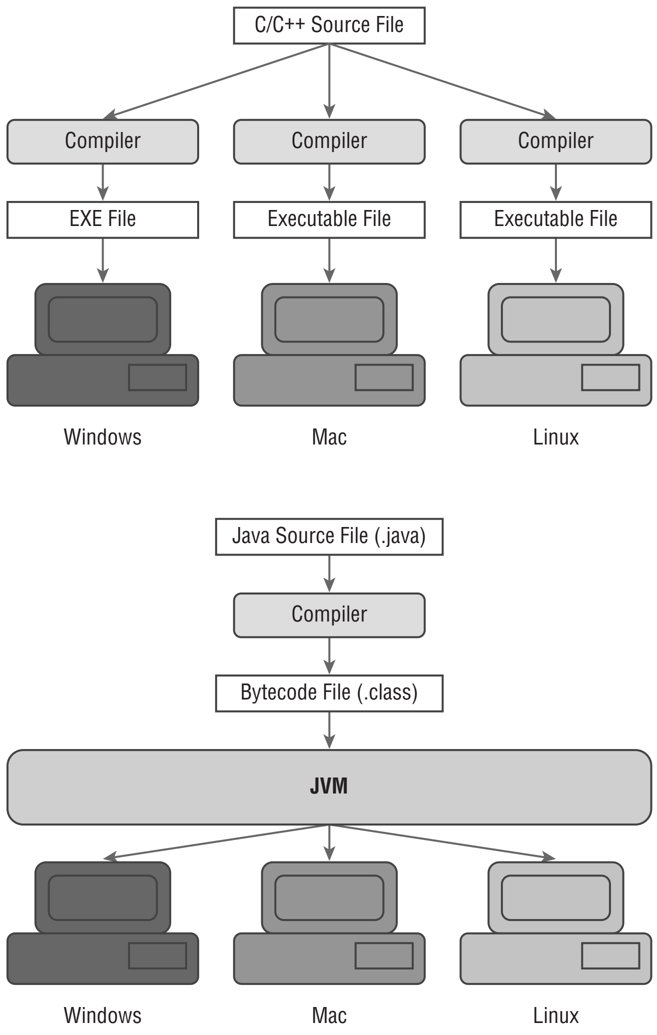


Figure 1.1: The conventional compilation process of the C/C++ programming language on different platforms (top) and the Java compilation process on different platforms (bottom)

1.2 Versions of Java

Java has had many versions; at the writing of this book, the current version is Java 11; by the time you read this, it will be Java 12. Alpha and Beta were the initial releases of the Java Development Kit (JDK) in 1995. JDK 1.0 was the first official version, released in 1996. Java JDK version 1.2 and newer are generally called Java 2. The collection of Java 2 languages, libraries, and tools is referred to as the Java 2 platform, or Java 2 Standard Edition (J2SE). Similarly, there are Java 5, Java 6, Java 7, and Java 8. The latest Java releases are Java 9 (July 2017), Java 10 (March 2018), Java 11 (September 2018), and Java 12 (March 2019). See Table 1.1 for details.

Table 1.1: Java Version History

VERSION	CODE NAME	RELEASE DATE	END OF LIFE
JDK Alpha and Beta		1995	Prior to 2008
JDK 1.0	Oak	January 1996	Prior to 2008
JDK 1.1		February 1997	Prior to 2008
J2SE 1.2	Playground	December 1998	Prior to 2008
J2SE 1.3	Kestrel	May 2000	Prior to 2008
J2SE 1.4	Merlin	February 2002	August 2008
J2SE 5.0	Tiger	September 2004	November 2009
Java SE 6	Mustang	December 2006	February 2013
Java SE 7	Dolphin	July 2011	April 2015
Java SE 8 (LTS)		March 2014	January 2019
Java SE 9		September 2017	March 2018
Java SE 10		March 2018	September 2018
Java SE 11 (LTS)		September 2018	
Java SE 12		March 2019	

For Java releases after Java SE 8, Oracle has designated a long-term-support (LTS) release every three years, and in between are non-LTS releases, also called *feature releases*, every six months. Java SE 9, Java SE 10, and Java 12 are all non-LTS releases, and Java SE 8 and Java SE 11 are LTS releases. Java end of life (EOL) occurs when the Java release is no longer publicly supported by Oracle. For the non-LTS releases, the EOL is the date of the next new release, and all the public support will be superseded. But for the LTS releases, the EOL is much longer, and customers will continue to get public support even after the new releases. That is why the widely used Java SE 8 has a much longer EOL than other releases. The next planned LTS release will be Java SE 17. This book is focused on the application of Java; the Java example codes used in this book will not be affected by the future Java releases.

For more information about Java releases and support road map, please visit the following:

```
https://www.oracle.com/technetwork/java/java-se-support-roadmap.html
https://en.wikipedia.org/wiki/Java_version_history
```

Each Java release is distributed as two different packages.

The **Java Runtime Environment (JRE)** is for running Java programs and is intended for end users. The JRE consists of the JVM and runtime libraries. You can use the JRE when you don't need to compile the Java program.

The **Java Development Kit (JDK)** is for software developers to compile, debug, and document Java programs. You will need to use the JDK in this book, as you will need to compile your Java programs.

1.3 Java Architecture

Figure 1.2 shows the relationship between the JDK, JRE, and JVM in the Java architecture. The JDK includes the JRE and Java development tools, and the JRE includes the JVM and library classes, as well as other files. Inside the JVM, there is a just-in-time (JIT) compiler, which compiles Java bytecode to native machine code during the execution of a Java program, that is, at run time. JIT improves the performance of Java applications.

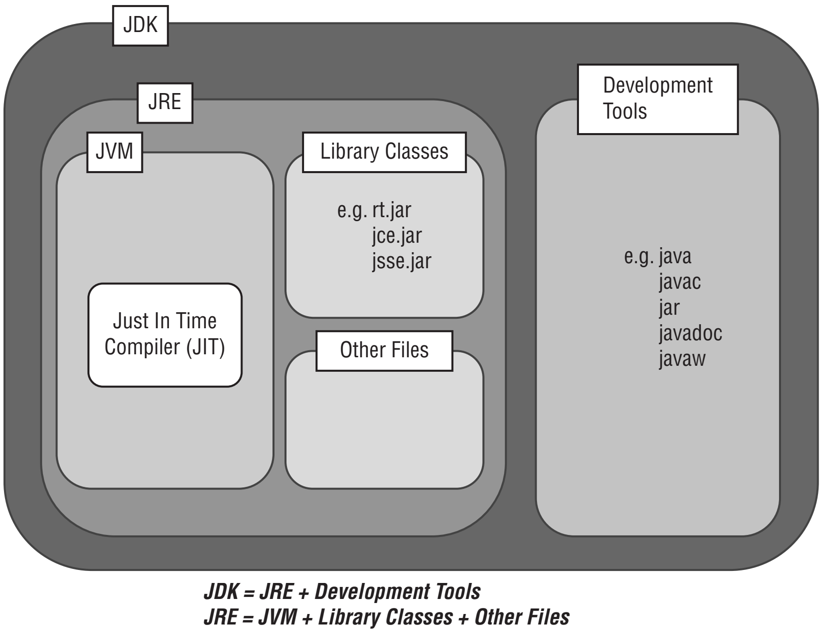
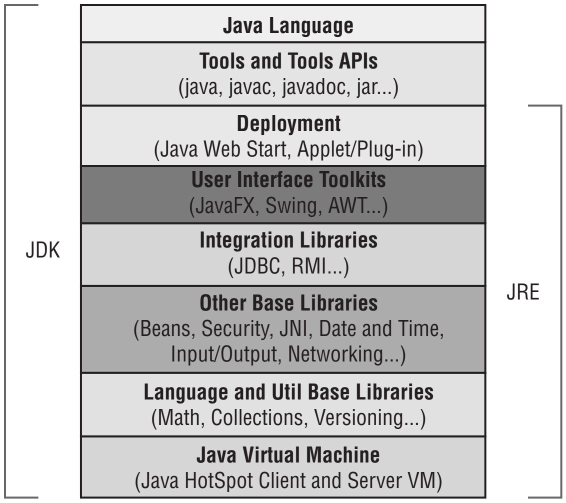


Figure 1.2: The relationship between the JDK, JRE, and JVM in Java architecture

Figure 1.3 shows a more detailed version of Java architecture; this was re-created from the original Oracle Java architecture diagram found here:

```
https://www.oracle.com/technetwork/java/javase/tech/index.html
```


Java Language
Tools and Tools APIs
(java, javac, javadoc, jar...)
Deployment
(Java Web Start, Applet/Plug-in)
User Interface Toolkits
(JavaFX, Swing, AWT...)
Integration Libraries
(JDBC, RMI...)
Other Base Libraries
(Beans, Security, JNI, Date and Time, Input/Output, Networking...)
Language and Util Base Libraries
(Math, Collections, Versioning...)
Java Virtual Machine
(Java HotSpot Client and Server VM)
JDK
JRE
Figure 1.3: A conceptual diagram of the Java architecture

1.4 Editions of Java

There are four Java platform editions.

Java Card for smartcards

Java ME (Micro Edition) for mobile devices

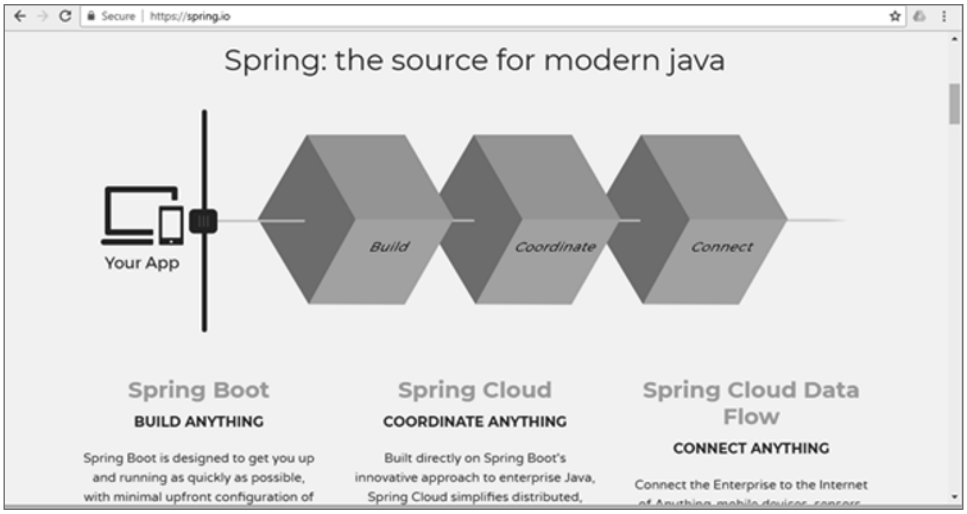
Java SE (Standard Edition) for standard personal computers

Java EE (Enterprise Edition) for large distributed enterprise or Internet environments

Java SE is what most people use for Java programming. This edition comes with the complete Java Class Library, which includes the basic types and objects, networking, security, databases, and the classic Swing graphical user interface (GUI) toolkit. Most versions also include the modern JavaFX toolkit, which is intended to replace the Swing GUI toolkit; however, starting with Java SE 11, the JavaFX toolkit is no longer included in the Java SDK and is redesigned as a separate, stand-alone library. This book will be mainly focused on Java SE.

1.5 The Java Spring Framework

Java Spring is the most popular development framework for creating Java enterprise applications. Java Spring is an open source framework. Initially written by Rod Johnson, it was released under the Apache 2.0 license in June 2003. One of the main advantages of the Spring framework is its layered architecture, which allows developers to select which of its components to use. Figure 1.4 shows the home page of the Java Spring Framework (`https://spring.io/`). Figure 1.5 shows the Guides page for the Framework (`https://spring.io/guides`).



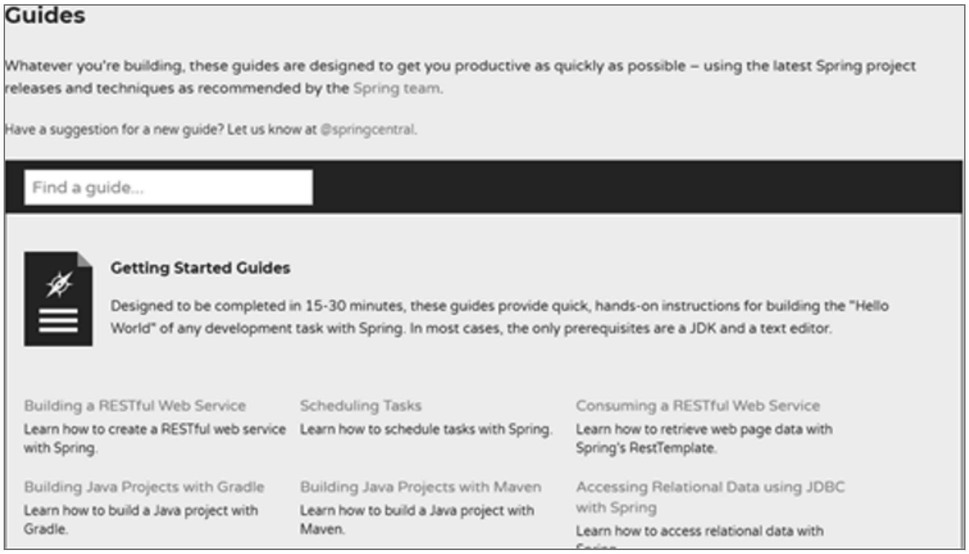
Spring: the source for modern java

Spring Boot
BUILD ANYTHING
Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration of

Spring Cloud
COORDINATE ANYTHING
Built directly on Spring Boot's innovative approach to enterprise Java, Spring Cloud simplifies distributed,

Spring Cloud Data Flow
CONNECT ANYTHING
Connect the Enterprise to the Internet

Figure 1.4: The home page of the Java Spring Framework



Guides

Whatever you're building, these guides are designed to get you productive as quickly as possible – using the latest Spring project releases and techniques as recommended by the Spring team.

Have a suggestion for a new guide? Let us know at @springcentral.

Getting Started Guides

Designed to be completed in 15-30 minutes, these guides provide quick, hands-on instructions for building the "Hello World" of any development task with Spring. In most cases, the only prerequisites are a JDK and a text editor.

Building a RESTful Web Service
Learn how to create a RESTful web service with Spring.

Scheduling Tasks
Learn how to schedule tasks with Spring.

Consuming a RESTful Web Service
Learn how to retrieve web page data with Spring's RestTemplate.

Building Java Projects with Gradle
Learn how to build a Java project with Gradle.

Building Java Projects with Maven
Learn how to build a Java project with Maven.

Accessing Relational Data using JDBC with Spring
Learn how to access relational data with

Figure 1.5: The Guides page in the Java Spring Framework

There are also several good Java Spring framework tutorials online.

```
https://www.tutorialspoint.com/spring/spring _ overview.htm
https://howtodoinjava.com/spring-5-tutorial/
https://java2blog.com/introduction-to-spring-framework/
```

1.6 Advantages and Disadvantages of Java

I've already noted some of Java's advantages, but it also has a few disadvantages that may affect your choice of a development language. This section provides a quick summary of both. Many items are the topics of chapters or sections later in this book.

1.6.1 Advantages

These are the advantages:

Free Cost Java is free to use, even for commercial applications, although you do need to pay for security and certain updates.

Simplicity Java is much easier to learn and to use than other programming languages. Java also uses automatic memory allocation and garbage collection.

Platform Independence Once compiled, Java programs can run on any operating system, thanks to the JVM.

Object Orientation Java is a fully object-oriented programming language that allows you to create reusable Java modules (classes). Chapter 3 introduces Java's object orientation.

Security Java is designed to be secure and safe. See Chapter 9 for information about security.

Multithreading With Java, you can easily develop multithreaded programs that run several tasks simultaneously. Chapter 3 also introduces multithreaded programming.

Networking Java provides a range of functions to make it easier to develop networking applications. Chapter 5 covers developing networking apps.

Mobile Development With Java, you can develop mobile applications, called *apps*, on Android systems. Chapter 6 covers developing apps for mobile devices.

Enterprise Development With Java, you can develop many enterprise applications, such as web servers and other application servers.

1.6.2 Disadvantages

These are the disadvantages:

Performance Java is much slower than other natively compiled languages, such as C or C++, because of the use of the JVM. Java also takes more memory space and has limited options for latency critical tuning.

GUI Development Generally speaking, it is not easy to develop GUI programs with Java, and the look and feel of the Java Swing toolkit is very different from native Windows, Mac, and Linux applications, although there are significant improvements in the JavaFX GUI toolkit. Chapter 4 shows how to overcome the difficulties and develop GUI apps using Java Swing and JavaFX.

1.7 Java Certification

Oracle offers a range of Java certificates, which can be generally divided into two levels, Associate and Professional, as shown in Figure 1.6 (`https://education.oracle.com/pls/web_prod-plq-dad/ou_product_category.getPageCert?p_cat_id=267`). You can start by applying for Java Foundations Certified Junior Associate, then move on to Oracle Certified Associate, and finally become an Oracle Certified Professional. Different Java versions require their own certificates. For example, there are separate certifications for Java SE 7 Programmer and Java SE 8 Programmer. Certificates for newer Java versions will continue to be introduced.

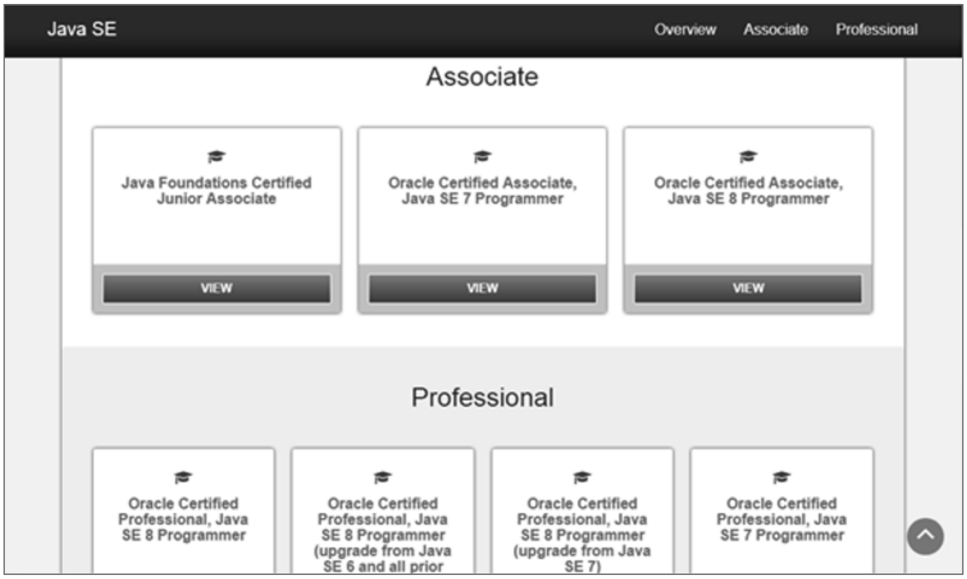

Java SE
Overview
Associate
Professional
Associate
Java Foundations Certified Junior Associate
VIEW
Oracle Certified Associate, Java SE 7 Programmer
VIEW
Oracle Certified Associate, Java SE 8 Programmer
VIEW
Professional
Oracle Certified Professional, Java SE 8 Programmer
Oracle Certified Professional, Java SE 8 Programmer (upgrade from Java SE 6 and all prior
Oracle Certified Professional, Java SE 8 Programmer (upgrade from Java SE 7)
Oracle Certified Professional, Java SE 7 Programmer

Figure 1.6: The Oracle Java Certification path

1.8 Summary

This chapter introduced the Java programming language, including its history, versions, and the four Java platform editions. It also introduced the popular Java Spring Framework for enterprise Java application development, summarized Java's advantages and disadvantages, and finally provided information about Java certification.

1.9 Chapter Review Questions

Q1.1. What is Java? Explain the difference between a Java source file and Java bytecode.

Q1.2. What is HotJava, and what is JavaScript?

Q1.3. What is platform independence?

Q1.4. Which Java versions are still supported?

Q1.5. Use a diagram to describe the Java architecture.

Q1.6. What are the JDK, JRE, JVM, and JIT?

Q1.7. What are the four Java platform editions?

Q1.8. What is the Java Spring Framework?

Q1.9. What are the advantages and disadvantages of Java?

Q1.10. What Java certifications are available?