Java Crash Course

Implementing core Java features, including data types, operators, and flow control mechanisms

Dr. Edward Lavieri Jr.

www.bpbonline.com

First Edition 2026

Copyright © BPB Publications, India

ISBN: 978-93-65896-237

Dedicated to

Java, not the language, but our late Tri-Color Pembroke Welsh Corgi. She was named Java for my fondness of the language and my wife's love of quality organic coffee. She was on my mind throughout the process of writing this book.

About the Author

Dr. Edward Lavieri Jr. is a software engineer and academic. He has developed software for nearly four decades. His projects include distributed, full-stack applications, mobile, desktop, and component apps. While a software engineer at his core, he also serves as a university dean at a university system which includes a college of computer science, a school of information technology, and a coding bootcamp, where he serves as the chief academic officer. He enjoys solving real-world business problems with technology.

Edward earned his doctorate in computer science from Colorado Technical University in 2014, holds three master's degrees and multiple postgraduate certificates, and is a senior member of the *Institute of Electrical and Electronics Engineers* (*IEEE*). He has published over a dozen technical books.

About the Reviewers

- **Dimakatso** is a software developer with a BSc degree in computer science with over 8 years of experience in software development. He specializes in using cutting-edge technologies, such as Spring/Quarkus for high-performance backends, Angular/React for intuitive user interfaces, and React Native for mobile development. His expertise extends across the full development lifecycle, from system design to deployment, ensuring seamless integration and user satisfaction.

- **Serhad** is a Java developer with 17 years of experience. He works extensively with Spring and the broader Java ecosystem, building scalable and maintainable software solutions. His professional background includes deep involvement in enterprise application development and system integration. Outside of work, he enjoys reading about history and exploring programming languages; always curious about the ideas that shape both technology and human progress.

Acknowledgement

I am very grateful to everyone who has contributed to the existence of this book.

I would like to thank my wife who was critical to my ability to maintain a work-authorship-life balance.

I am also very grateful to BPB Publications for trusting me with this book. Thank you for your guidance and support in every step of writing this book.

I would also like to acknowledge the reviewers, technical experts and editors who helped shape the book into its final form. Your insights and expertise have undoubtedly enhanced the quality of this book.

Preface

Java is one of the most prevalent programming languages today, powering everything from enterprise applications to mobile platforms. Whether you are just starting your programming journey or transitioning to Java from another language, Java Crash Course is your essential guide to mastering this powerful language.

This book takes you step-by-step from the basics of Java syntax, data types, and control structures, through core **object-oriented programming** (**OOP**) concepts, to advanced topics like concurrency, streams, and **graphical user interfaces** (**GUIs**). Along the way, you will learn how to handle exceptions, work with collections, connect to databases, and gain experience with modern Java features such as lambdas and functional programming. Hands-on exercises, real-world projects, and clear examples can reinforce your understanding and provide practical skills you can apply immediately.

By the end of this book, you should not only have a deep understanding of Java, but also the confidence to build scalable, efficient applications and tackle real-world coding challenges with ease. Whether you are a student, developer, or IT professional, this crash course can equip you with the skills to succeed in your career.

Chapter 1: Introduction to Java Programming- This chapter introduces the Java programming language and its ecosystem. The chapter begins with a brief history of Java, how it has evolved, and how it has maintained relevance in the modern software development industry. The benefits of learning Java have never been greater, especially in today's world of enterprise applications, cloud computing, and mobile development. This chapter also walks you through setting up your development environment, including installing the **Java Development Kit** (**JDK**) and configuring an **integrated development environment** (**IDE**). Finally, you will have the opportunity to write and run your first Java program, giving you hands-on experience with the language from the very start.

Chapter 2: Core Java Concepts- This chapter explores the foundational concepts that every Java programmer must master. Starting with data types and variables, the chapter covers how Java handles information, followed by a look at operators and control structures that form the basis of decision-making and logic in programs. The chapter explores how to write methods, understanding parameters, return types, and scope, which are essential for building reusable code. By the end of this chapter, you should have a solid grasp of core Java syntax and functionality and be ready to tackle more complex problems.

Chapter 3: Object-oriented Programming Basics- This chapter introduces OOP, one of the key paradigms that make Java a versatile and powerful language. Coverage includes the fundamental OOP principles of encapsulation, inheritance, polymorphism, and abstraction. The chapter also covers how to define and use classes and objects, manage constructors and overloading, and explore the super keyword to enhance code reusability and maintainability. Through examples and hands-on practice, this chapter shows you how to apply these principles to design robust, scalable software applications.

Chapter 4: Advanced OOP Concepts- Building on the basics of OOP, this chapter explores advanced topics like interfaces and abstract classes, which provide flexibility and extensibility in Java design. The chapter demonstrates how to implement multiple inheritance using interfaces and explores functional interfaces and default methods, which are powerful modern Java tools. The chapter also covers lambdas and method references, which simplify functional programming in Java, making your code cleaner and more concise. By the end of this chapter, you should have a deeper understanding of how to structure large, complex applications with an object-oriented approach.

Chapter 5: Handling Errors and Exceptions- Effective error handling is crucial for creating robust Java applications, and this chapter introduces you to Java's sophisticated exception handling system. The chapter explores the difference between checked and unchecked exceptions, how to use try-catch-finally blocks, and how to throw and handle exceptions in your code. Additionally, you can learn how to create custom exceptions and implement best practices for handling errors. The aim of this chapter is to help ensure you can handle unexpected issues gracefully and build applications that fail safely and informatively.

Chapter 6: Data Structures and Collections- This chapter explores data structures, beginning with arrays and array lists, and extending into the **Java Collections Framework** (**JCF**). Coverage includes how to work with lists, sets, maps, and queues, helping you to understand when to use each. The chapter also covers sorting and searching collections and the role of iterators and enhanced for loops. You can learn how to compare objects using Comparable and Comparator interfaces and leverage Java's utility classes to simplify collection management. Mastering these structures can help enable you to write efficient, scalable code for handling large datasets.

Chapter 7: Concurrency and Multithreading- Concurrency is a vital aspect of modern software, and this chapter introduces you to multithreading in Java. The chapter starts by explaining how to create and manage threads using the Thread class and the Runnable interface. From there, the chapter explores synchronization, locks, and the Executor framework, which simplify concurrent programming. The chapter also covers Java's concurrency utilities like

CountDownLatch and CyclicBarrier, giving you tools to manage complex thread interactions. Best practices for writing thread-safe code are emphasized to help you avoid common pitfalls in concurrent programming.

Chapter 8: Streams and Functional Programming- This chapter focuses on functional programming in Java, particularly the Streams API introduced in Java 8. This chapter explains how to work with streams, performing operations like filtering, mapping, and reducing data in a functional style. The chapter also explores functional interfaces such as Predicate, Consumer, and Supplier, which power much of modern Java's functional capabilities. Hands-on opportunities are provided for working with lambdas and parallel streams to handle large datasets efficiently. By mastering these techniques, you can be better prepared to write cleaner, more efficient code.

Chapter 9: Input/Output- File handling is essential for many Java applications, and this chapter introduces you to Java's I/O systems. The chapter explains how to read and write files using the java.nio package and explores streams for handling both character and byte data. The chapter also covers serialization and deserialization for storing and retrieving objects from files. Buffered and data streams are introduced for optimizing performance, giving you the skills to manage files efficiently and securely in your applications.

Chapter 10: Database Connectivity- This chapter covers connecting Java applications to databases using **Java Database Connectivity** (**JDBC**). The chapter shows how to setup JDBC in your projects, establish connections to databases, and execute SQL queries. The chapter also explains how to use prepared statements and callable statements for interacting with databases securely and efficiently. Additionally, the chapter explores how to handle database transactions in Java, ensuring consistency and reliability in your applications. Best practices for database operations are emphasized to ensure optimal performance.

Chapter 11: GUI Programming- GUIs allow users to interact with applications visually, and this chapter introduces you to JavaFX, Java's modern GUI framework. The chapter explains how to setup JavaFX in your environment and work with the scene graph to create visually appealing layouts. The chapter also covers handling user events, creating controls like buttons and text fields, and binding properties to enhance your application's interactivity. By the end of this chapter, you should be able to build simple, functional GUI applications in Java.

Chapter 12: Modern Java Features- Java has evolved significantly over the years, and this chapter explores the most important recent Java features. You will have the opportunity to work with streams and lambdas, explore Project Jigsaw's modules, and dive into features from the latest Java releases like JEP 477 (implicitly declared classes and instance main methods).

This chapter also includes best practices for modern Java development, covering the latest tools and techniques for writing efficient, maintainable code.

Chapter 13: Debugging, Testing, and Deployment- Once our Java code is written, we must be able to debug, test, deploy, and support our applications. This chapter shows you how to debug Java programs using IDE tools and logging. You can learn how to write unit tests using JUnit and apply **test-driven development** (**TDD**) principles to ensure your code is reliable. Finally, the chapter covers packaging and deploying Java applications using build tools like Maven and Gradle, helping to prepare you to release production-ready software.

Chapter 14: Real-world Java Projects- This chapter provides you the opportunity to apply the skills you learned in earlier chapters to real-world Java projects. This starts by building a simple command-line application, then progressing to a database application using JDBC and a JavaFX-based GUI application.You will have the opportunity to tackle a multithreaded data processor and write a functional program using streams and lambdas. These projects are designed to help reinforce your understanding of key Java concepts while giving you practical experience that can be applied in real-world development scenarios.

Chapter 15: Conclusion and Next Steps- The final chapter recaps the key concepts covered in the book and provides guidance on how to continue learning Java beyond this book. The chapter includes advice on preparing for industry certifications, along with resources to further deepen your knowledge. The chapter also looks ahead to trends in Java development, helping to ensure you stay current in this fast-evolving field.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/5ab372

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Java-Crash-Course.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at: **errata@bpbonline.com**

Your support, suggestions and feedback are highly appreciated by the BPB Publications' Family.

Table of Contents

Chapter 1
Introduction to Java Programming

Introduction

In this chapter, we will be introduced to Java and its family of tools, libraries, frameworks, and other resources. We will briefly look at Java's creation and then document its evolution over the years. We will also understand why Java remains relevant today as a powerful application development suite for enterprise applications, distributed systems, cloud-based systems, mobile applications, embedded systems, and more.

This chapter will highlight why understanding Java can benefit modern software developers and engineers. We will mainly cover setting up a Java development environment and providing a guide to help you write and run your first Java application.

Structure

This chapter covers the following topics:

- Overview of Java
- History and evolution of Java
- Importance of Java
- Java's relevance to modern software development

- Setting up the development environment
- Writing and running your first Java program

Objectives

By the end of this chapter, we will cover Java's history and evolution and describe the significance of learning Java. We will also setup a development environment for programming with Java, write and execute a simple Java program, and understand Java syntax.

Overview of Java

Java is a high-level programming language introduced in 1995 and continues to be one of the most popular languages today. This popularity is due to a myriad of reasons, including its readability, object-oriented nature, available libraries and tools, scalability, and more. It is characterized as a high-level language because humans and computers can read Java code. Another reason for Java’s wide use is that it is a cross-platform language. This means you can write Java code on your computer (regardless of whether it is a Mac, PC, or Linux machine), and your compiled code can be run on any device that has a **Java Virtual Machine** (**JVM**) installed. This represents a tremendous advantage for developers.

In the following sections, we will discuss Java’s key features, architecture, and platforms in detail.

Key features

Java is the programming language of choice because of its plethora of features. Let us look at some of these features as follows:

- **Platform independence**: When we program in Java, we create text files with the .java extension. Once we complete our application, we compile our code into **bytecode**, which can be executed on any device with a JVM installed. This is perhaps the most revered feature of the Java programming language, as we do not need to be concerned about the executing computer’s **operating system** (**OS**) or environment. Java allows us to focus on coding our application and not worry about troublesome porting operations to multiple systems.
- **Object-oriented programming (OOP)**: Java is an object-oriented language, which means that objects represent both behavior and our application’s data. This programming approach helps us create modular, scalable, and maintainable applications. As covered in *Chapter 3, Object-oriented Programming Basics*, the primary OOP principles are abstraction, encapsulation, inheritance, and polymorphism. Java’s implementation of OOP makes it easy for developers and is, therefore, one of Java’s key features.

- **Garbage collection (GC)**: Java includes an automatic memory management system called GC that automatically deallocates memory occupied by application data that is no longer required. Having this automated for us is a tremendous time saver. We will cover GC in *Chapter 4, Advanced OOP Concepts*, when we talk about the object lifecycle. The automated and efficient manner of GC makes this a key feature worth studying.
- **Security focus**: Java is referred to as a secure language due to its secure runtime environment. The JVM does a wonderful job of verifying our bytecode (remember, that is the intermediate file between our code and the JVM) to ensure it does not include malicious code. Java has additional security-related features, including a security **application programming interface** (**API**) that enables us to use encryption, authentication, and more. The security focus of Java makes it an attractive choice when selecting programming languages.
- **Standard library**: Java has an extensive standard library, which gives us a robust set of utilities that we can include in our programs without having to code them. This includes mathematical functions, **graphical user interfaces** (**GUIs**), file handling, networking, database connectivity, and so much more. This impressive library is a key reason why Java is so popular. We will use several utilities from the standard library throughout this book.
- **Multithread support**: In *Chapter 7, Concurrency and Multithreading*, we will have the opportunity to explore Java's native multithreading support. This allows us to program our applications to run multiple operations simultaneously (concurrently). This lets us take advantage of modern systems with multiple cores. As you will see, Java includes an Executor framework that we can use to control our thread pools, resulting in greater efficiency and smarter resource management.

A review of these six features provides insight into the robustness of the Java programming language.

Java architecture

The three core components of Java's architecture are the **Java Development Kit** (**JDK**), JVM, and the **Java Runtime Environment** (**JRE**). The definitions for each of these components are as follows:

- **JDK**: We use the JDK to develop Java applications; it is a requirement. The JDK includes the JRE (explained in the subsequent points) and the tool that compiles our Java code into bytecode.
- **JVM**: The JVM executes this bytecode (compiled Java code). As we mentioned earlier, the JVM is responsible for making Java a platform-independent language. It achieves this by acting as an intermediary between the host computer's OS and the Java

application. The JVM's intermediary role involves abstracting away any hardware and OS specifics and translating the bytecode (platform-independent) into machine code.

- **JRE**: The JRE is the third core component of the Java programming language. It contains the JVM and libraries that allow us to execute our Java programs.

Java platforms

Java offers four platforms; each designed for unique development purposes. The first edition is the **Java Standard Edition** (**Java SE**). This version offers core libraries and APIs that are foundational to all Java programs. This standard edition is extremely powerful and is used for coding examples throughout this book.

The **Java Enterprise Edition** (**Java EE**) platform can be considered the next step up from Java SE. Java EE includes libraries that support enterprise-level application development, including large-scale systems, multi-tiered applications, and secure and scalable network applications.

The **Java Micro Edition** (**Java ME**) platform is a subset of Java SE, as it is designed to run on devices that lack computing, memory, and storage resources evident in larger systems. The JVM contained in this edition has a small footprint. Typical deployments include embedded systems and **Internet of Things** (**IoT**) devices.

The fourth platform is **JavaFX**, which is used for creating dynamic internet applications. One benefit of using JavaFX is that the resultant applications use hardware acceleration to support graphic rendering. We will cover JavaFX in *Chapter 11, GUI Programming*.

History and evolution of Java

Java celebrated its 30th birthday on May 23, 2025. Introduced by *Sun Microsystems* in 1995, Java aimed to achieve platform independence by writing software once and running it on a wide range of different hardware platforms. Originally called **Oak**, Java was transformed from a predominantly embedded systems language into a cross-platform language ready to support the growth of internet technology.

Rapid adoption

Software developers quickly adopted the Java platform to create web-based applications, and as Java improved, it started being used for enterprise and large-scale systems. Java 2 was introduced in the late 1990s and introduced the Java SE, Java EE, and Java ME described in the previous section. This made the language applicable to all developers regardless of whether they were developing desktop computers, creating large-scale enterprise systems, creating applications for mobile devices, or coding embedded systems. Java catapulted to the most frequently used programming language for new applications.

Oracle years

Approximately 15 years after Java was introduced, it was acquired by *Oracle* in October 2010. Oracle has been a great steward of Java, as we have seen it continue to evolve and keep pace with contemporary development needs. They implemented a six-month release schema, resulting in new features being continually released. This helps keep Java relevant for developers who have choices. There are several hundred programming languages in use today. Java has been in the top five of most lists since its initial release.

Ready for the future

Today, Java continues to be one of the most popular programming environments used in multiple industries, including academia, business, communications, finance, healthcare, and more. It continues to evolve and has an extremely active developers' community. Java's ongoing evolution will keep pace with our modern development needs.

Importance of Java

Java is one of the most widely used programming languages for applications of all types, including embedded systems, mobile applications, desktop solutions, large-scale enterprise systems, distributed systems, and more. This popularity is different from why you should learn Java; what makes Java so popular is why learning to program with Java is worth your time.

Here are some reasons why Java could be a good choice for your development needs:

- You can develop with Java using a system running Windows, macOS, or Linux.
- Java is platform-independent, so you can develop once on your computer and deploy your app everywhere.
- Java is object-oriented, which makes it easy for us to implement abstraction, encapsulation, inheritance, and polymorphism. More on those concepts in *Chapter 3, Object-oriented Programming Basics.*
- Java supports modern concepts such as modularity, multithreading, streams, and more.
- Java includes automatic memory deallocation with its garbage collector. This saves us tremendous time from having to program memory deallocation into our apps.
- The number of tools, libraries, and extensions to the Java language is tremendous, as is the developer community.

In the subsequent sections, you will see that getting started with Java is easy. We can download Java SE and the JDK at no cost and use them inside a free **integrated development environment** (**IDE**), so the startup cost is nonexistent.

Beyond learning Java to solve current programming requirements, Java developers are in great demand. Search any job board, and you will see multiple listings for Java developers and software engineers with Java knowledge.

Java's relevance to modern software development

Java is highly relevant to modern software development and is the language of choice for countless developers who create and maintain applications across industries. Developers are typically required to develop systems that can be deployed on multiple devices. Java's platform independence makes this easy, saving developers and product teams time and frustration.

One of the reasons Java is so relevant for modern software development is its robust ecosystem, which includes tools, libraries, and frameworks that streamline and shorten development time. This ecosystem lets us easily create systems that are scalable, modular, and maintainable.

New paradigms

As new paradigms and requirements are introduced, Java continues to answer the call. For example, the need to support functional programming was implemented in Java with the use of lambda expressions and the Stream API. As you will see in *Chapter 8, Streams and Functional Programming*, this is an essential advancement for dealing with large-scale systems, big data, and cloud computing. Java makes it easy to develop modular systems and implement a microservice architecture; both are key to modern software development.

Security

Security is paramount to virtually all software development projects when privacy and data protection are important, such as in finance, government, and healthcare applications. Java is a type-safe language, which means that it only permits operations to be performed on specified data types.

Paramount to the security of our applications at runtime is how data in memory is managed. As you will read about in *Chapter 4*, *Advanced OOP Concepts*, Java has automated processes to free up data in memory when it is no longer needed by the application. This process, GC, is not only a security feature but also contributes to overall system performance.

Java has APIs for cryptography, **public key infrastructure** (**PKI**), authentication, and secure communications. Furthermore, several modules of the JDK contain security APIs. As you will see in *Chapter 3*, *Object-oriented Programming Basics*, Java's object-oriented approach allows us to abstract and encapsulate data, intensifying the secure nature of Java applications.

Setting up the development environment

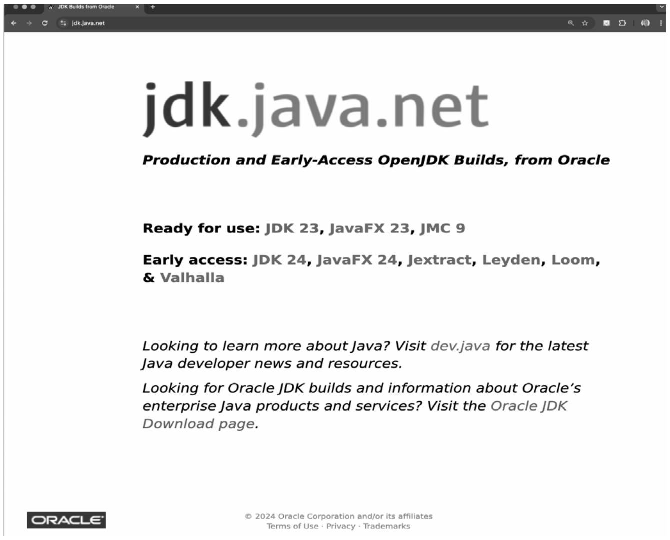
To setup your computer for Java development, the following steps are involved:

1. Download and install the JDK
2. Setup environment variables
3. Choosing and configuring an IDE

Let us get started with the first step.

Downloading and installing the JDK

Oracle maintains an OpenJDK website at **https://jdk.java.net**. This site will list the available JDK versions. You can click on the latest JDK version listed in the **Ready for use** area. As shown in *Figure 1.1*, that version, at the time of this writing, is **JDK 23**:



jdk.java.net

Production and Early-Access OpenJDK Builds, from Oracle

Ready for use: JDK 23, JavaFX 23, JMC 9

Early access: JDK 24, JavaFX 24, Jextract, Leyden, Loom, & Valhalla

Looking to learn more about Java? Visit dev.java for the latest Java developer news and resources.

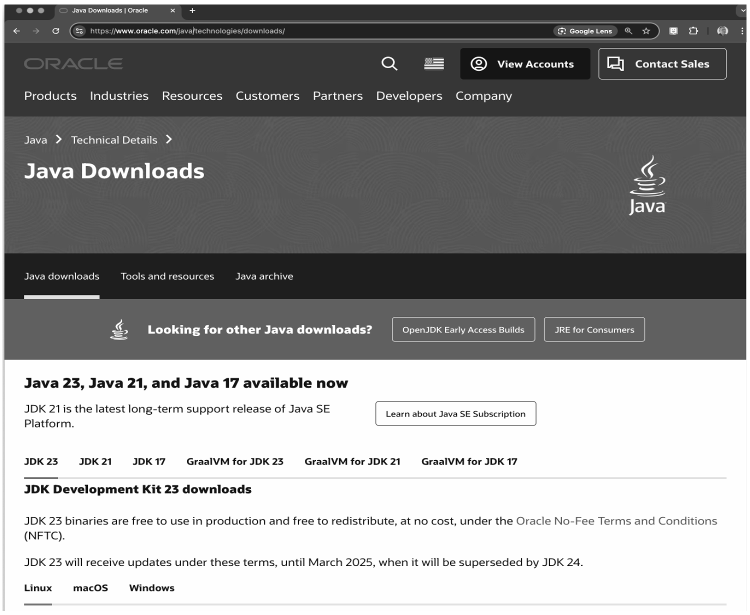
Looking for Oracle JDK builds and information about Oracle's enterprise Java products and services? Visit the Oracle JDK Download page.

© 2024 Oracle Corporation and/or its affiliates
Terms of Use · Privacy · Trademarks

***Figure 1.1**: OpenJDK page showing JDK 23 as the latest version*

Note: You can also visit Oracle's JDK download page to download the latest JDK: https://www.oracle.com/java/technologies/downloads/

The OpenJDK page provides a concise list of available versions and a link to Oracle's official download page, shown in *Figure 1.2.* You can see the available JDKs on that page, starting with JDK 23 as the latest. That is the one we will download and install. At the bottom of *Figure 1.2*, you can see tabs for **Linux**, **macOS**, and **Windows**. Select the tab for your computer's OS. With JDK 23 and your OS tab selected, you will see download options. Select the one that is specific to your computer, as shown in the following figure:


Java Downloads | Oracle

https://www.oracle.com/java/technologies/downloads/

ORACLE

View Accounts

Contact Sales

Products Industries Resources Customers Partners Developers Company

Java > Technical Details >

Java Downloads

Java downloads Tools and resources Java archive

Looking for other Java downloads? OpenJDK Early Access Builds JRE for Consumers

Java 23, Java 21, and Java 17 available now

JDK 21 is the latest long-term support release of Java SE Platform.

Learn about Java SE Subscription

JDK 23 JDK 21 JDK 17 GraalVM for JDK 23 GraalVM for JDK 21 GraalVM for JDK 17

JDK Development Kit 23 downloads

JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.

Linux macOS Windows

***Figure 1.2**: Oracle's official JDK download page*

Follow the on-screen instructions to complete the JDK installation process.

Note: You installed it on your computer, as you will need that information in the next section.

Setting up environment variables

Environment variables are user-defined values that we can employ to ensure our system recognizes our Java installation. The following are the instructions specific to your OS:

1. **In Windows**:
    a. Open the **System Properties**.
    b. Select the **Advanced** tab.
    c. Click on **Environmental Variables**.
    d. Add a new system variable named **`JAVA_HOME`** and set the value to the JDK installation path.
    e. Go to the *Verify your installation* section below.
2. **In MacOS or Linux**:
    a. Open a Terminal window.

b. Enter the following in the Terminal window: **`export JAVA_HOME=/actual-path/`**.

   Note: replace `/actual-path/` with the file path to your JDK 23 `bin` folder. For example: `export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin`.

c. Enter the following in the Terminal window: **`export PATH=$JAVA_HOME/bin:$PATH`**.

d. Go to the *Verify Your Installation* section below.

Verify your installation

You are now ready to verify your installation. Open a terminal window (macOS or Linux) or a command prompt (Windows) and enter the **`java -version`** to ensure your system recognizes JDK 23.

The output will be as follows:

```
java version "23" 2024-09-17
Java(TM) SE Runtime Environment (build 23+37-2369)
Java HotSpot(TM) 64-Bit Server VM (build 23+37-2369, mixed mode, sharing)
```

Next, we will verify that our system recognizes the Java compiler. Enter **`javac -version`**. Your output should reflect the following:

```
javac 23
```

Now that we have installed JDK 23, created our environmental variable, and verified our installation, we are ready to select, download, and configure an IDE.

Choosing and configuring an IDE

Now that we have successfully installed Java on our machine, we must choose an IDE. IDEs are software applications that we use to develop software. They provide rich source code editing, code completion, debugging tools, and more. There are several IDEs available to us. The following are the major ones:

- Eclipse
- IntelliJ IDEA
- NetBeans
- **Visual Studio Code** (**VS Code**)

Your IDE is up to you and will not impact your final Java application. Much like driving to a store, it does not matter if you take your car, minivan, or truck; they all get you to your destination. So, IDE selection is up to the developer. Of course, the IDE might be mandated if you are working on a team or in a company. This book uses VS Code because it is easy to use.

Downloading and installing VS Code

VS Code's download page is at **https://code.visualstudio.com/Download**. Visit that page and select the version specific to your OS. As shown in *Figure 1.3*, there are **Windows**, **Linux**, and **macOS** versions:

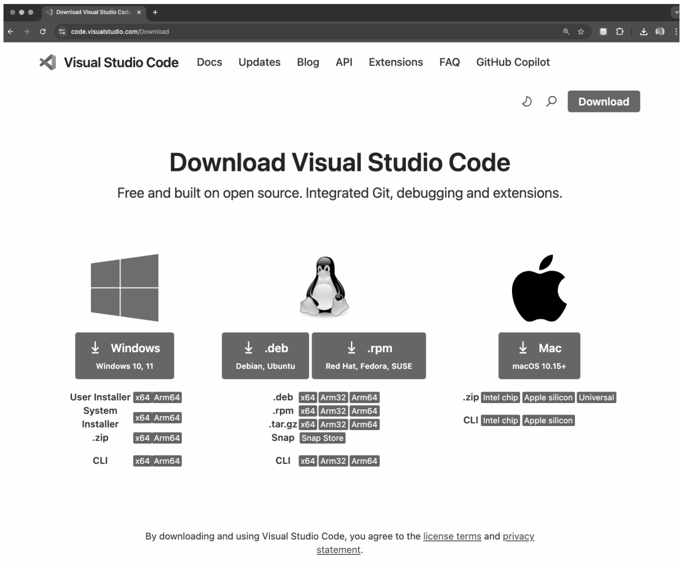

code.visualstudio.com/Download
Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
GitHub Copilot
Download
Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.
Windows
Windows 10, 11
.deb
Debian, Ubuntu
.rpm
Red Hat, Fedora, SUSE
Mac
macOS 10.15+
User Installer x64 Arm64
System Installer x64 Arm64
.zip x64 Arm64
CLI x64 Arm64
.deb x64 Arm32 Arm64
.rpm x64 Arm32 Arm64
.tar.gz x64 Arm32 Arm64
Snap Snap Store
CLI x64 Arm32 Arm64
.zip Intel chip Apple silicon Universal
CLI Intel chip Apple silicon
By downloading and using Visual Studio Code, you agree to the license terms and privacy statement.

Figure 1.3: *VS Code download page*

Once your download is complete, follow the installation instructions.

Installing the Java extension pack

Our next step is to install the Java extension pack in VS Code. VS Code is not just for Java. The base installation of VS Code supports CSS, HTML, JavaScript, TypeScript, and more without needing extensions. With extensions, VS Code can support C, C++, C#, Python, Java, and more.

Open VS Code and click the *extensions* icon at the bottom of the left navigation panel in *Figure 1.4*. Using the search bar, locate the **Extension Pack for Java** by *Microsoft*, as follows:

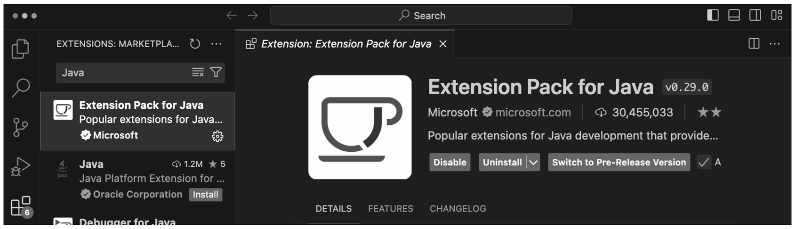

Search
EXTENSIONS: MARKETPLA...
Java
Extension Pack for Java
Popular extensions for Java...
Microsoft
Java 1.2M ★ 5
Java Platform Extension for ...
Oracle Corporation Install
Debugger for Java
Extension: Extension Pack for Java
Extension Pack for Java v0.29.0
Microsoft microsoft.com 30,455,033 ★★
Popular extensions for Java development that provide...
Disable
Uninstall
Switch to Pre-Release Version
DETAILS
FEATURES
CHANGELOG

Figure 1.4: *Top section of VS Code extensions area with Extension Pack for Java selected*