MODERN JAVA PROGRAMMING

FROM FIRST PROGRAM TO PRODUCTION-READY SYSTEMS WITH JAVA 26

WRITE BETTER CODE
Clean, modern, and maintainable code

BUILD REAL APPLICATIONS
Practical examples that scale

DESIGN FOR PERFORMANCE
Optimize and build high-performance apps

SHIP RELIABLE SYSTEMS
Best practices for production-ready code

ROWAN LEE JACOB

Modern Java Programming

From First Program to Production-Ready Systems with Java 26

Steve Publications

This book is available at https://leanpub.com/modernjavaprogramming

This version was published on 2026-07-31

Contents

From First Program to Production-Ready Systems with Java 26

This book takes you from zero Java knowledge to advanced, production-ready proficiency using modern Java. You will install a complete development environment, master the language from syntax to virtual threads, build real applications with testing and observability, and learn how to deploy them in containers. Every code example is complete, idiomatic, and compiles exactly as written with Java 26.

Introduction: Why Java Still Matters

The Numbers That Tell the Story

In 2026, Java remains one of the most widely used programming languages in the world. It powers everything from massive e-commerce platforms and banking systems to Android applications and embedded devices. According to consistent industry surveys including TIOBE and Stack Overflow, Java has held a top-five position for over two decades. More than 3 million developers use it globally, and it runs on billions of devices.

These numbers are not an accident. Java endures because it solves real problems reliably: it scales to handle millions of concurrent requests, its tooling ecosystem is unmatched in maturity, and its long-term support releases provide stability for organizations that cannot afford constant disruption.

The language has also evolved significantly. The Java that shipped in 2014 with lambda expressions looks nothing like the Java available today. Records eliminate boilerplate data classes. Pattern matching simplifies complex type inspection. Virtual threads make it practical to write highly concurrent code using straightforward, blocking-style logic instead of callback chains or reactive frameworks.

From Applets to Virtual Threads

Java launched in 1996 as part of Sun Microsystems' "write once, run anywhere" vision. The early years were defined by the Java Virtual Machine and the promise of platform independence. Java 5 (2004) introduced generics, annotations, and enums. Java 8 (2014) brought lambda expressions and the Stream API, opening the door to functional-style programming [9].

Since then, the release cadence accelerated to every six months, with long-term support releases every two years [10]. Java 17 (2021) was an LTS release that added sealed classes and text blocks. Java 21 (2023) included virtual threads, record patterns, and pattern matching for switch as permanent

features. Java 25 became the next LTS release in September 2025, finalizing scoped values [41], flexible constructor bodies [42], compact source files [43], and module imports [44]. This book uses Java 26 [1], released in March 2026, which adds HTTP/3 support to the standard library [45], improves garbage collection performance [50], and continues refining preview features like structured concurrency [46] and lazy constants [48].

This rapid evolution has not come at the cost of stability. Existing Java code continues to run unchanged across versions. The language adds capabilities without breaking what already works, which is why enterprises trust it with mission-critical systems.

What This Book Covers (and Doesn't)

This book assumes you understand basic programming concepts such as variables, loops, and functions from some language. It does not assume any prior Java knowledge. You will learn:

- How to install and configure a modern Java development environment
- The complete Java syntax: types, operators, control flow, methods, classes
- Object-oriented design principles with practical examples
- Modern Java features: records, sealed classes, pattern matching, lambdas, streams
- Concurrency from traditional threads through virtual threads
- JVM internals, garbage collection, and performance tuning
- Building production-ready applications with testing, logging, metrics, and security
- Packaging, containerization, and deployment

This book does not cover frameworks like Spring Boot in depth. Frameworks are important tools, but they layer on top of the language and platform. Understanding Java itself first makes any framework easier to learn. Where frameworks or libraries are used (for JSON processing, database access, testing), I explain why they are chosen and how they integrate with the core language.

Your First Milestone

By the end of Chapter 1, you will have installed a JDK, created your first Maven project, and run a Java program. Starting in Chapter 4, you will build a Library Management System that evolves throughout the book: first as plain Java objects, then with JPA persistence, HTTP endpoints, automated tests, observability, security, and containerized deployment. By Chapter 17, you will understand how to write highly concurrent services using virtual threads. By the final chapter, you will know what it takes to make a Java application production-ready.

Versions Used in This Book

Every code example in this book uses the following explicitly stated versions. You should match these versions when following along to ensure everything compiles and runs as shown.

- **JDK:** Java 26 (OpenJDK build 26.0.x or later)
- **Build tool:** Apache Maven 3.9.16
- **Testing framework:** JUnit Jupiter 5.10.2
- **Mocking library:** Mockito 5.23.0
- **JSON processing:** Jackson 2.20.x
- **Database connection pool:** HikariCP 7.1.0
- **Persistence API:** Jakarta Persistence API 3.2 with Hibernate ORM 6.6.x
- **Logging:** SLF4J 2.0.18 with Logback 1.6.0
- **Metrics:** Micrometer 1.17.0
- **Observability:** OpenTelemetry Java SDK 1.64.0
- **Container testing:** Testcontainers 2.0.5
- **Property-based testing:** jqwik 1.10.x

If you use a different version of any tool, behavior may differ. The book notes where specific features depend on particular versions.

Chapter 1: Setting Up Your Java Development Environment

Before writing any code, you need a working development environment. This chapter walks you through installing the Java Development Kit, choosing and configuring a build tool, and running your first program. Every step includes commands you can execute directly.

Choosing Your JDK

Java is specified by the Java SE (Standard Edition) specification, but multiple vendors provide implementations. All major implementations are built from OpenJDK, the open-source reference project. The key differences between distributions lie in support policies, additional tools, and licensing.

The most common choices are:

Distribution	Maintainer	Best for
Eclipse Temurin	Eclipse Foundation (Adoptium)	General use; vendor-neutral default
Amazon Corretto	Amazon	AWS-heavy environments
Oracle JDK	Oracle	Enterprise support contracts
Azul Zulu	Azul Systems	Commercial support, wide platform coverage
GraalVM	Oracle	Polyglot programming, native image compilation

For this book, Eclipse Temurin is recommended as the default choice [11]. It is free under open-source licensing, passes the Java Technology Compatibility Kit (TCK) tests that verify compliance with the specification, and is the most widely adopted community distribution. If you are already using Amazon Corretto or another vendor's build, any of them will work identically for the code examples.

All distributions provide the same core language features. A program written for Eclipse Temurin 26 runs unchanged on Amazon Corretto 26, Oracle JDK 26, or any other compliant OpenJDK-based distribution.

Installing JDK 26

You need Java 26 specifically for this book. Earlier versions lack features covered here, including HTTP/3 support and the latest refinements to structured concurrency and pattern matching. If you are using an LTS-focused workflow, Java 25 LTS is also suitable but will miss some preview features discussed in later chapters.

Linux (APT-based)

On Ubuntu, Debian, or similar distributions:

```
# Install SDKMAN! for easy JDK management
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"

# Install Java 26 via SDKMAN
sdk install java 26.0.2-tem
sdk use java 26.0.2-tem
```

Alternatively, using the system package manager:

```
sudo apt update
sudo apt install -y openjdk-26-jdk
```

macOS

Using Homebrew:

```
brew install openjdk@26
```

Then configure your shell to use it by adding these lines to your .zshrc or .bash_profile:

```
export JAVA_HOME=$(/usr/libexec/java_home -v 26)
export PATH="$JAVA_HOME/bin:$PATH"
```

Or using SDKMAN! (same commands as Linux above).

Windows

Download the installer from Eclipse Temurin or use Chocolatey:

```
choco install temurin26
```

After installation, set the JAVA_HOME environment variable to your JDK installation path and add %JAVA_HOME%\bin to your PATH.

Verifying Your Installation

Regardless of platform, verify that Java 26 is installed and accessible:

```
java -version
javac -version
```

Expected output:

```
openjdk version "26.0.x" 202X-XX-XX
OpenJDK Runtime Environment (build 26.0.x+...)
OpenJDK 64-Bit Server VM (build 26.0.x+..., mixed mode, sharing)
```

The `java` command runs Java programs. The `javac` command compiles Java source code. Both must be present and report version 26.

Selecting a Build Tool

Java projects rarely rely on a single source file. They include multiple classes, external dependencies (libraries), configuration files, and tests. A build tool automates compilation, dependency management, testing, and packaging. The two dominant options are Maven and Gradle.

Maven uses XML-based configuration with a conventional directory structure. Its "convention over configuration" approach means most projects look similar. Dependencies are declared in a pom.xml file using group IDs, artifact IDs, and versions. Maven has a large plugin ecosystem and is straightforward to understand once you know the basics.

Gradle uses Groovy or Kotlin DSL for configuration, offering more flexibility. It can be faster for large projects due to incremental builds and caching. Gradle's flexibility comes with complexity: build scripts can become intricate, and there are many ways to accomplish the same thing.

For learning Java, Maven is recommended. Its explicit structure makes it easier to understand what is happening at each step. The code examples in this book use Maven exclusively.

Installing Maven

Maven requires a JDK but does not include one. Verify your Java installation first, then install Maven [12,13].

Linux (APT-based)

```
sudo apt update
sudo apt install -y maven
```

Or using SDKMAN!:

```
sdk install maven 3.9.16
```

macOS

Using Homebrew:

```
brew install maven
```

Windows

Using Chocolatey:

```
choco install maven
```

Verifying Maven

Check that Maven is installed and reports the correct version:

```
mvn -v
```

Expected output includes lines like:

```
Apache Maven 3.9.16
Maven home: /path/to/maven
Java version: 26.0.x, vendor: Eclipse Adoptium, runtime: /path/to/java
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "6.x", arch: "amd64"
```

The important parts are that Maven version is at least 3.9.x and it detects your Java 26 runtime.

Creating Your First Project

Maven projects follow a standard directory structure. You can create one manually or use the Maven archetype plugin to generate a template. Let us start with the manual approach so you understand the layout.

Create a new directory for your project:

```
mkdir -p ~/projects/hello-java/src/main/java/com/example
cd ~/projects/hello-java
```

The directory structure means:

- `src/main/java` holds your application's Java source files
- Packages are represented by subdirectories matching the package name (com/example here)
- Maven expects this layout by convention

Create the pom.xml file in the project root. This is Maven's Project Object Model that defines your project's metadata and dependencies:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>hello-java</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Hello Java</name>
    <description>First Maven project for Modern Java Programming</description>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>26</maven.compiler.release>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.13.0</version>
                <configuration>
                    <release>26</release>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Key elements:

- `groupId`, `artifactId`, `version` uniquely identify your project. Together they form a coordinate like com.example:hello-java:1.0.0-SNAPSHOT.
- `maven.compiler.release` set to 26 tells the compiler to generate bytecode compatible with Java 26 and enforce that only Java 26 APIs are used.
- The maven-compiler-plugin configuration ensures consistent compilation settings.

Now create your first Java source file at src/main/java/com/example/HelloJava.java:

```
package com.example;

public class HelloJava {
    public static void main(String[] args) {
        System.out.println("Hello from Java 26!");
        System.out.println("JVM version: " + Runtime.version());
    }
}
```

This file declares a class named HelloJava in the com.example package. The main method is the entry point for any standalone Java application. It must be public, static, return void, and accept a String array parameter.

Running Your First Program

Maven can compile and run your project with a single command. From the project root directory:

```
mvn compile exec:java -Dexec.mainClass="com.example.HelloJava"
```

The first time you run this, Maven downloads required plugins from the central repository. On subsequent runs, they are cached locally.

Expected output (truncated for clarity):

```
[INFO] Scanning for projects...
[INFO] Building Hello Java 1.0.0-SNAPSHOT
[INFO] --- maven-compiler-plugin:3.13.0:compile ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file with javac [debug release 26] to target/classes
[INFO] --- exec-maven-plugin:3.5.0:java ---
Hello from Java 26!
JVM version: 26.0.x+...
[INFO] BUILD SUCCESS
```

What happened:

1. Maven scanned pom.xml and understood the project configuration.
2. The compiler plugin invoked javac to compile HelloJava.java into a class file at target/classes/com/example/HelloJava.class.
3. The exec plugin ran the compiled class using java com.example.HelloJava.

You can also run the program directly with the java command after compilation:

```
mvn compile
java -cp target/classes com.example.HelloJava
```

The `-cp` flag specifies the classpath, which is where Java looks for compiled classes. Here, target/classes is the directory where Maven placed the compiled output.

Understanding the Project Structure

After running mvn compile, your project directory looks like this:

```
hello-java/
├── pom.xml
├── src/
│   └── main/
│       └── java/
│           └── com/
│               └── example/
│                   └── HelloJava.java
└── target/
    └── classes/
        └── com/
            └── example/
                └── HelloJava.class
```

The src directory contains your source code. The target directory is generated by Maven and contains compiled output. You should never edit files in target manually; they are recreated each time you build.

For larger projects, the structure expands to include tests and resources:

```
project/
├── pom.xml
├── src/
│   ├── main/
│   │   ├── java/          # Application source code
│   │   └── resources/     # Configuration files, templates, etc.
│   └── test/
│       ├── java/          # Test source code
│       └── resources/     # Test-specific resources
└── target/                # Build output (generated)
```

This convention is followed by every Maven project. Once you learn it, navigating any Java project becomes straightforward.

The Master Project for This Book

Rather than creating separate projects for each chapter, this book uses a single multi-module Maven project that grows throughout the text. All code examples fit into this structure, and every listing includes its exact file path within it. You can follow along by maintaining one project and adding code as you progress.

Create the root directory and parent POM:

```
mkdir -p ~/projects/modern-java-book
cd ~/projects/modern-java-book
```

File: pom.xml (parent POM)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>modern-java-book</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <name>Modern Java Programming - Master Project</name>
    <description>Multi-module project containing all examples from the
    ↪  book</description>

    <modules>
        <module>basics</module>
        <module>oop</module>
        <module>design</module>
        <module>collections</module>
        <module>functional</module>
        <module>modern-features</module>
        <module>annotations</module>
        <module>datetime</module>
        <module>io-http</module>
        <module>database</module>
        <module>concurrency</module>
        <module>virtualthreads</module>
        <module>jvm-internals</module>
        <module>testing</module>
        <module>observability</module>
        <module>security</module>
        <module>deployment</module>
        <module>interop</module>
        <module>production-readiness</module>
        <module>library-system</module>
    </modules>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>26</java.version>

        <!-- Dependency versions -->
```

```
        <junit-jupiter.version>5.10.2</junit-jupiter.version>
        <mockito.version>5.23.0</mockito.version>
        <assertj.version>3.26.3</assertj.version>
        <jackson.version>2.20.0</jackson.version>
        <slf4j.version>2.0.18</slf4j.version>
        <logback.version>1.6.0</logback.version>
        <micrometer.version>1.17.0</micrometer.version>
        <opentelemetry.version>1.64.0</opentelemetry.version>
        <testcontainers.version>2.0.5</testcontainers.version>
        <jqwik.version>1.10.1</jqwik.version>
        <hikaricp.version>7.1.0</hikaricp.version>
        <hibernate.version>6.6.9.Final</hibernate.version>
        <jakarta-persistence.version>3.2.0</jakarta-persistence.version>
        <h2.version>2.3.232</h2.version>
        <postgresql.version>42.7.5</postgresql.version>
        <swagger.version>2.2.28</swagger.version>
        <spotless.version>2.44.3</spotless.version>
        <jacoco.version>0.8.12</jacoco.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <!-- Testing -->
            <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter</artifactId>
                <version>${junit-jupiter.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.mockito</groupId>
                <artifactId>mockito-core</artifactId>
                <version>${mockito.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.assertj</groupId>
                <artifactId>assertj-core</artifactId>
                <version>${assertj.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>net.jqwik</groupId>
                <artifactId>jqwik</artifactId>
                <version>${jqwik.version}</version>
                <scope>test</scope>
            </dependency>

            <!-- JSON -->
            <dependency>
```

```
            <groupId>com.fasterxml.jackson</groupId>
            <artifactId>jackson-bom</artifactId>
            <version>${jackson.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

        <!-- Logging -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>${logback.version}</version>
        </dependency>

        <!-- Metrics -->
        <dependency>
            <groupId>io.micrometer</groupId>
            <artifactId>micrometer-bom</artifactId>
            <version>${micrometer.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

        <!-- Observability -->
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-bom</artifactId>
            <version>${opentelemetry.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

        <!-- Database -->
        <dependency>
            <groupId>com.zaxxer</groupId>
            <artifactId>HikariCP</artifactId>
            <version>${hikaricp.version}</version>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>${jakarta-persistence.version}</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate.orm</groupId>
```

```
            <artifactId>hibernate-core</artifactId>
            <version>${hibernate.version}</version>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>${h2.version}</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>${postgresql.version}</version>
            <scope>runtime</scope>
        </dependency>

        <!-- Testcontainers -->
        <dependency>
            <groupId>org.testcontainers</groupId>
            <artifactId>testcontainers-bom</artifactId>
            <version>${testcontainers.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.13.0</version>
                <configuration>
                    <release>${java.version}</release>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.5.2</version>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>3.5.0</version>
            </plugin>
            <plugin>
                <groupId>org.jacoco</groupId>
```

```
                    <artifactId>jacoco-maven-plugin</artifactId>
                    <version>${jacoco.version}</version>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

Each module has its own pom.xml that inherits from this parent. For example, the basics module:

File: basics/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.example</groupId>
        <artifactId>modern-java-book</artifactId>
        <version>1.0.0-SNAPSHOT</version>
    </parent>

    <artifactId>basics</artifactId>
    <name>Basics Module</name>

    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Throughout the book, every code listing specifies its file path relative to this structure. For example:

- `basics/src/main/java/com/example/basics/Variables.java`
- `oop/src/main/java/com/example/oop/Book.java`
- `library-system/src/main/java/com/example/library/model/Book.java`

To build all modules from the root:

```
cd ~/projects/modern-java-book
mvn clean install
```

To build a single module:

```
cd ~/projects/modern-java-book/basics
mvn clean install
```

This unified structure ensures every example compiles with its declared dependencies, and you can progressively add code as you work through the book.

Summary

You now have a working Java development environment with JDK 26 and Maven installed. You created a Maven project, wrote a simple program, compiled it, and ran it. The next chapter explains how Java source code becomes a running program: the compilation process, bytecode, and the Java Virtual Machine.

Chapter 2: How Java Works - Compilation and Execution

Understanding how Java programs are compiled and executed is essential for writing good code. The model differs from languages that compile directly to machine code, and these differences have practical consequences for how you structure your programs, debug them, and reason about performance.

From Source Code to Bytecode

When you write Java code in a .java file, it is human-readable source code. Before it can run, the Java compiler (javac) translates it into bytecode, an intermediate representation that runs on the Java Virtual Machine rather than directly on your computer's processor.

Run javac manually to see the compilation step:

```
cd ~/projects/hello-java
javac -d target/classes src/main/java/com/example/HelloJava.java
```

The `-d` flag specifies the output directory. After running this, inspect the generated class file:

```
ls -la target/classes/com/example/
```

You will see HelloJava.class. This is not a text file; it is a binary format containing bytecode instructions and metadata about the class. You can view the bytecode using the javap disassembler:

```
javap -c target/classes/com/example/HelloJava.class
```

Output:

```
Classfile /path/to/target/classes/com/example/HelloJava.class
  Last modified ...; size 938 bytes
  MD5 checksum ...
  Compiled from "HelloJava.java"
public class com.example.HelloJava
  minor version: 0
  major version: 70
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
Constant pool:
   #1 = Methodref          #4.#15         // java/lang/Object."<init>":()V
   #2 = Fieldref           #16.#17        //
    ↪  java/lang/System.out:Ljava/io/PrintStream;
   #3 = String             #18            // Hello from Java 26!
   #4 = Class              #19            // java/lang/Object
   #5 = Methodref          #20.#21        //
    ↪  java/lang/Runtime.version:()Ljava/lang/Runtime$Version;
   #6 = String             #22            // JVM version:
   #7 = Methodref          #23.#24        //
    ↪  java/io/PrintStream.println:(Ljava/lang/String;)V
  ...
public static void main(java.lang.String[]);
    descriptor: ([Ljava/lang/String;)V
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #2                  // Field
         ↪  java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #3                  // String Hello from Java 26!
         5: invokevirtual #7                  // Method
         ↪  java/io/PrintStream.println:(Ljava/lang/String;)V
         8: getstatic     #2                  // Field
         ↪  java/lang/System.out:Ljava/io/PrintStream;
        11: ldc           #6                  // String JVM version:
        13: invokestatic  #5                  // Method
        ↪  java/lang/Runtime.version:()Ljava/lang/Runtime$Version;
        16: invokestatic  #25                 // Method
        ↪  java/lang/String.valueOf:(Ljava/lang/Object;)Ljava/lang/String;
        19: invokedynamic #26,  0             // InvokeDynamic
        ↪  #0:makeConcatWithConstants:(Ljava/lang/String;Ljava/lang/Runtime$Version;)Ljava/
        24: invokevirtual #7                  // Method
        ↪  java/io/PrintStream.println:(Ljava/lang/String;)V
        27: return
```

The bytecode instructions (getstatic, ldc, invokevirtual, etc.) are not processor-specific machine code. They are operations defined by the JVM specification. The major version 70 corresponds to Java 26 (the formula is: Java version + 44, so 26 + 44 = 70).

Bytecode serves as a platform-independent intermediate format. The same

.class file compiled on Linux runs on macOS or Windows without recompilation, because each platform's JVM interprets and executes the bytecode instructions.

The Java Virtual Machine

The Java Virtual Machine (JVM) is the runtime environment that executes Java bytecode. It is not a single program but a specification implemented by multiple vendors. HotSpot, developed by Oracle and Sun Microsystems, is the most widely used JVM implementation and ships with Eclipse Temurin, Amazon Corretto, and Oracle JDK.

The JVM performs three essential functions:

1. **Loading classes**: When your program runs, the JVM loads the required class files into memory using a class loader subsystem. It resolves references between classes, ensuring that every type your code uses is available.
2. **Executing bytecode**: The JVM's execution engine processes bytecode instructions. Modern HotSpot uses a just-in-time (JIT) compiler: initially, bytecode is interpreted for fast startup, but frequently executed ("hot") code is compiled to native machine code on the fly for better performance.
3. **Managing memory**: The JVM automatically allocates and reclaims memory through garbage collection. When objects are no longer reachable by any running code, the garbage collector identifies and frees their memory without explicit programmer intervention.

These responsibilities happen behind the scenes, but understanding them helps when debugging issues like OutOfMemoryError, slow startup times, or unexpected behavior from class loading.

Class Files and the Classpath

A class file contains the bytecode for one type: a class, interface, enum, or annotation. The filename matches the public type name with a .class extension. Inner classes have encoded names like OuterClass$InnerClass.class.

When the JVM runs a program, it needs to find the required class files. The classpath is the set of locations (directories and JAR files) where the JVM searches for classes. By default, the classpath includes only the current directory (.).

You specify the classpath using:

- The `-cp` or `-classpath` flag when running java
- The CLASSPATH environment variable (rarely recommended in practice)
- Build tools like Maven, which compute the classpath automatically

When you ran `java -cp target/classes com.example.HelloJava`, the classpath was set to target/classes. The JVM found HelloJava.class there and also found classes from the Java standard library (like System, String, Runtime) in its internal bootstrap classpath.

JAR files are essentially ZIP archives containing .class files and resources. When you add a dependency like Jackson for JSON processing, Maven downloads the corresponding JAR and includes it on the classpath during compilation and execution. The JVM treats directories and JAR files equivalently when searching for classes.

The Main Method

Every standalone Java application has an entry point: a method with the exact signature:

```
1  public static void main(String[] args)
```

Each part matters:

- `public`: The JVM can access this method from outside the class.
- `static`: The method belongs to the class itself, not to any instance. The JVM calls it before creating any objects.
- `void`: The method does not return a value to the operating system.
- `main`: This is the name the JVM looks for by convention.
- `String[] args`: Command-line arguments passed to the program, provided as an array of strings.

When you run `java com.example.HelloJava first second third`, the args array contains ["first", "second", "third"]. You can process these inside main:

```
package com.example;

public class Greet {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Hello, world!");
        } else {
            for (String name : args) {
                System.out.println("Hello, " + name + "!");
            }
        }
    }
}
```

Save this as src/main/java/com/example/Greet.java and run:

```
mvn compile
java -cp target/classes com.example.Greet Alice Bob
```

Output:

```
Hello, Alice!
Hello, Bob!
```

A class can have only one main method. If you want multiple entry points, create separate classes, each with its own main method.

Write Once, Run Anywhere

The "write once, run anywhere" slogan describes Java's platform independence: compile your code once into bytecode, and it runs on any system with a compatible JVM. This is not marketing hype; it is how Java works at the architectural level.

The bytecode format is defined by the Java Virtual Machine Specification, and all compliant JVMs implement the same instruction set. A .class file

compiled on a Linux machine with Eclipse Temurin runs identically on a Windows machine with Amazon Corretto, provided both are targeting the same Java version.

There are practical caveats:

- File paths use different separators on different operating systems (\ on Windows, / on Unix). Use java.nio.file.Path instead of string concatenation for portability.
- Line endings differ (\r\n on Windows, \n on Unix). Most APIs handle this automatically, but it matters when reading raw files.
- Some native libraries (used via JNI) are platform-specific and must be provided for each target OS.

For pure Java code that uses only standard library APIs, these issues rarely arise. The JVM abstracts away the underlying operating system, and your code runs the same everywhere.

Summary

Java programs go through a two-step execution model: source code is compiled to bytecode by javac, and bytecode is executed by the JVM. The JVM loads classes from the classpath, interprets or JIT-compiles bytecode, and manages memory automatically. The main method serves as the application entry point with a fixed signature. This architecture enables platform independence: the same compiled code runs on any system with a compatible JVM.

In the next chapter, you will start writing real Java code, learning the language syntax from the ground up: variables, types, operators, control flow, and methods.

Chapter 3: Language Fundamentals - Syntax, Types, and Control Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Variables and Primitive Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Integer Overflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Strings and Immutability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

StringBuilder for Efficient Concatenation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Operators and Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Arithmetic Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Relational and Equality Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Logical Operators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Control Flow Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

If-Else Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Switch Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Loops

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Arrays and Iteration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 4: Object-Oriented Programming in Java

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Classes and Objects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Constructors and Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Flexible Constructor Bodies (Java 25+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Encapsulation and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Inheritance and Polymorphism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Interfaces and Abstraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Abstract Classes vs. Interfaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Building the Library Management System Domain Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Core Entities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Service Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Repository Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

In-Memory Implementation (for now)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Application Entry Point

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 5: Designing with Objects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Cohesion and Coupling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Example: Low Cohesion (Bad)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Example: High Cohesion (Good)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

SOLID Principles in Java

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Single Responsibility Principle (SRP)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Open/Closed Principle (OCP)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Liskov Substitution Principle (LSP)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Interface Segregation Principle (ISP)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Dependency Inversion Principle (DIP)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Composition Over Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Immutability Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Builder Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Design Mistakes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 6: Packages, Modules, and Project Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Package Declaration and Naming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Naming Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Import Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Static Imports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Module Import Declarations (Java 25+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Java Platform Module System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

When to Use Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Organizing a Maven Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Standard Single-Module Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Multi-Module Projects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 7: Exceptions and Error Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Checked vs. Unchecked Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Try-Catch-Finally Blocks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Try-With-Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Creating Custom Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Exception Handling Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

1. Catch What You Can Handle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

2. Preserve the Cause

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

3. Use Specific Exception Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

4. Fail Fast

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

5. Document Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Logging Errors Properly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 8: Generics and Type Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Why Generics Exist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Generic Classes and Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Type Parameters and Bounds

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Wildcards and the PECS Principle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Type Erasure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Generic Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Utility Methods with Generics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Generic Comparators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 9: The Collections Framework

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Collection Interfaces Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Lists in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

ArrayList

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

LinkedList

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Creating Lists Concisely

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Sets and Hashing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

HashSet

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The hashCode and equals Contract

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

TreeSet and LinkedHashSet

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Maps and Key-Value Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

HashMap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

TreeMap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

ConcurrentHashMap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Queues and Deques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

PriorityQueue

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

ArrayDeque

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Collection Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 10: Functional Programming with Lambdas and Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Lambda Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Functional Interfaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Method References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Stream API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Stream Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Filtering and Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Reducing and Aggregating

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Collecting Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Performance Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 11: Modern Java Features - Records, Sealed Classes, Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Record Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Records vs Traditional Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Sealed Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Pattern Matching for instanceof

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Switch Expressions and Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Exhaustive Matching with Sealed Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Record Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Primitive Types in Patterns (Preview, Java 23+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Lazy Constants (Preview, Java 26)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Text Blocks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Version Compatibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 12: Annotations and Reflection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Built-in Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Defining Custom Annotations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Processing Annotations at Runtime

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Reflection Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

When to Use (and Avoid) Reflection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Annotation Processors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 13: Date, Time, and Locale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Legacy Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Core Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Durations and Periods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Time Zones

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Formatting and Parsing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 14: File I/O, Networking, and HTTP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The NIO.2 File API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Reading and Writing Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JSON Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

HTTP Client

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Building a Simple HTTP Service

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Library Management System REST API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Resource Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 15: Database Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JDBC Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Connection Pooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Parameterized Queries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Transaction Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Introduction to JPA/Hibernate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Entity Relationships and the N+1 Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Mapping the Library Domain with JPA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Many-to-One Relationships

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

One-to-Many Relationships

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Many-to-Many Relationships

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The N+1 Query Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Solutions to N+1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Choosing the Right Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Cascade Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

LazyInitializationException

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Testing Database Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 16: Concurrency Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Threads and Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Shared State and Race Conditions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Atomic Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Thread Pools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Concurrent Collections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

CompletableFuture: Composable Asynchronous Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Basic Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Combining Multiple Futures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Error Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

ForkJoinPool for CPU-Bound Parallelism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 17: Virtual Threads and Structured Concurrency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Platform Thread Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Virtual Threads Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Creating Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

When to Use Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Thread Pinning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Scoped Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Structured Concurrency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Timeout Handling with onTimeout() (Java 26+)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Migration Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

From Platform Thread Pools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

From Reactive Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Things to Watch For

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Applying Virtual Threads to the Library Management System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 18: JVM Internals and Memory Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JVM Architecture Overview

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Class Loading: How the JVM Finds and Loads Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Delegation Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Bootstrap, Platform, and Application Loaders

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Loading Process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Custom Class Loaders

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Class Loading Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Heap and Stack Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Heap Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Stack Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Garbage Collection Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

GC Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

G1GC (Default)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

ZGC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Generational ZGC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Shenandoah

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Tuning the JVM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Heap Size Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Container-Aware JVM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Useful JVM Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Production Recommendations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Memory Leaks in Java

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Diagnosing Memory Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JVM Troubleshooting Guide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

OutOfMemoryError: Java heap space

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

OutOfMemoryError: Metaspace

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

High CPU Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

High GC Pause Times

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Thread Deadlocks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Slow Application Startup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Connection Pool Exhaustion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Using jcmd Effectively

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 19: Testing Java Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Test Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JUnit 5

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Parameterized Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Mocking with Mockito

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Integration Testing with Testcontainers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Property-Based Testing with jqwik

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Testing the Library Management System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Unit Tests for Domain Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Unit Tests for Repository Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Integration Tests with Testcontainers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

End-to-End API Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Test Coverage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 20: Logging, Metrics, and Observability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Structured Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Metrics Collection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Distributed Tracing with OpenTelemetry

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Running a Local OpenTelemetry Collector

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Health Checks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Alerting Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Integrating Observability into the Library Management System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 21: Security in Java Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Common Vulnerabilities: OWASP Top 10

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Injection Prevention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Broken Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Cryptographic Failures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Security Misconfiguration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Vulnerable and Outdated Components

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Input Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Authentication and Authorization Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Secure Configuration and Secret Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Environment Variables (Development and Simple Deployments)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

HashiCorp Vault Integration (Production)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Running Vault Locally for Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Cloud Provider Secret Managers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Secret Management Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Principle of Least Privilege

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Defense in Depth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Cryptography APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 22: Packaging, Deployment, and Containerization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Building JARs and Uber-JARs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Native Images with GraalVM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Dockerizing Java Applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JVM in Containers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Deployment Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

CI/CD Pipeline Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 23: Interoperability and Legacy Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Calling Legacy Java Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

javax.* to jakarta.* Migration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Removed APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Migrating from Older Java Versions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java 8 to Java 11

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java 11 to Java 17

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java 17 to Java 21

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java 21 to Java 25

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java 25 to Java 26

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Java and Other Languages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JNI (Java Native Interface)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

JNA (Java Native Access)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Foreign Function & Memory API (Project Panama)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

XML Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

DOM Parsing (for small documents)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Jackson XML (for object mapping)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Maintaining Backward Compatibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Chapter 24: Production Readiness Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Code Quality Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Performance Baselines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Reliability Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Documentation Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Production Readiness Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Continuous Improvement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Capstone: The Complete Production-Ready Library Management System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Project Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

The Production Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Dockerfile for Production Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

What This Capstone Demonstrates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Conclusion: Your Next Steps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

What You Have Built

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Where Java Is Heading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Learning Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/modernjavaprogramming.