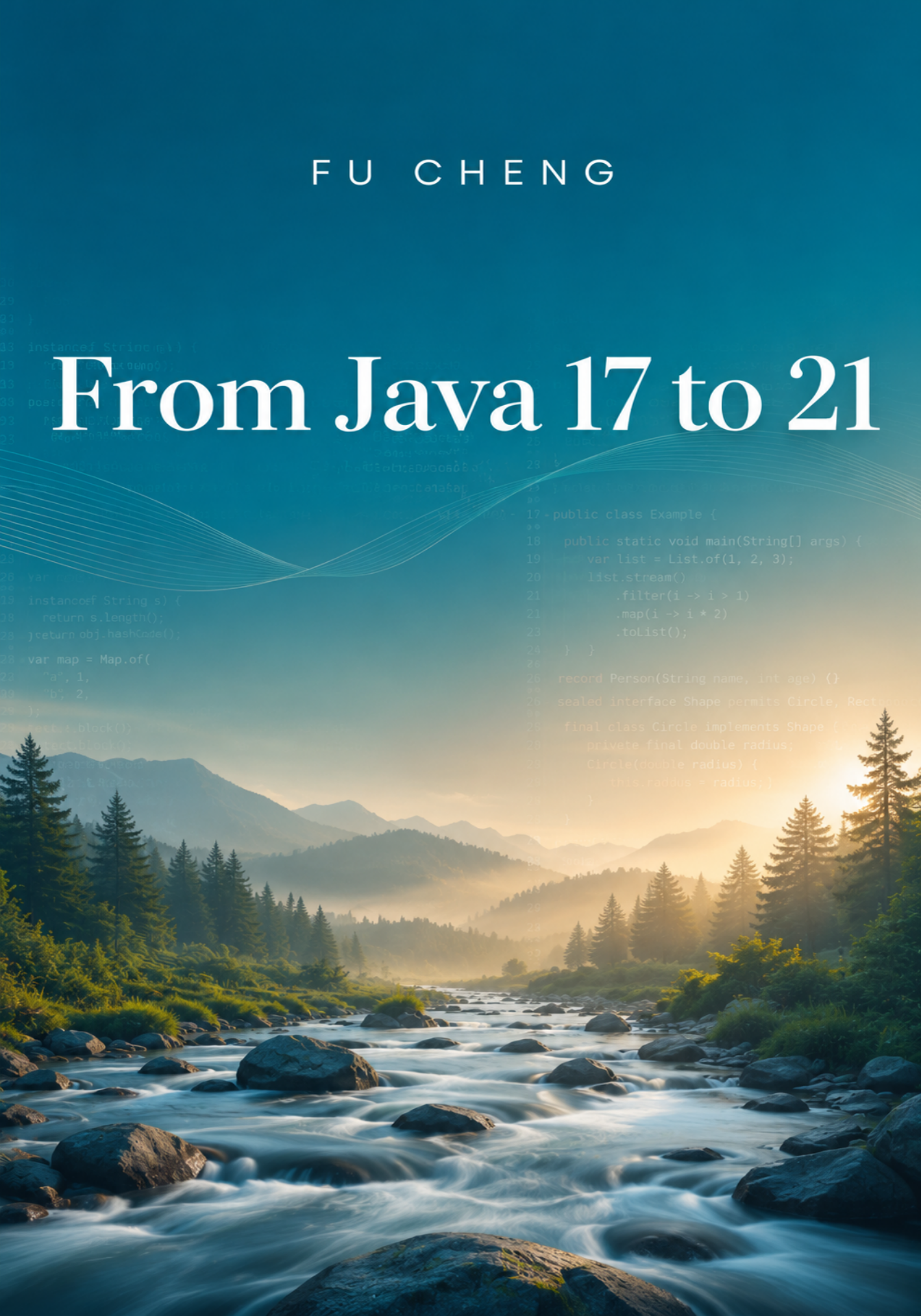
FU CHENG

# From Java 17 to 21



# From Java 17 to Java 21

## Upgrade to Java 21 LTS from Java 17 LTS

Fu Cheng

This book is available at https://leanpub.com/java17to21

This version was published on 2026-10-20




This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2026 Fu Cheng

## Also By Fu Cheng

Exploring Java 25
Text-to-SQL, Spring AI Implementation with RAG
Understanding Java Virtual Threads
From Java 21 to Java 25
Hands-on Spring AI
Build Native Java Apps with GraalVM
From Java 11 to Java 17
ES6 Generators
A Practical Guide for Java 8 Lambdas and Streams
Lodash 4 Cookbook
JUnit 5 Cookbook

# Table of Contents

**Introduction** . . . . . **1**
JEPs in Different Java Releases . . . . . 1
Install Java 21 . . . . . 3
Build Tools . . . . . 4
Source Code . . . . . 4

## Java Language . . . . . 5

**String Templates** . . . . . **6**
Template Expressions . . . . . 6
String Template . . . . . 6
Template Processor . . . . . 6
SMS Message Template . . . . . 6
Format Processor . . . . . 6

**Unnamed Classes and Instance Main Methods** . . . . . **7**
Java Program Launch Protocol . . . . . 7
Unnamed Classes . . . . . 7

**Code Snippets in Java API Documentation** . . . . . **8**
Inline Snippets . . . . . 8
External Snippets . . . . . 8
Markup Tags . . . . . 8
Tag References . . . . . 9

**Pattern Matching** . . . . . **10**
Record Patterns . . . . . 10
Pattern Matching for `switch` . . . . . 10
Unnamed Patterns and Variables . . . . . 11

# Core Library . . . . . 12

**UTF-8 By Default** . . . . . **13**

**Internet-Address Resolution SPI** . . . . . **14**

**Simple Web Server** . . . . . **15**

**Sequenced Collections** . . . . . **16**
- `SequencedCollection` . . . . . 16
- `SequencedSet` . . . . . 16
- `SequencedMap` . . . . . 16
- Classes Hierarchy . . . . . 16

**Reimplement Core Reflection with Method Handles** . . . . . **17**

**Key Encapsulation Mechanism API** . . . . . **18**

# Multi-threading . . . . . 19

**Virtual Threads** . . . . . **20**
- Create Virtual Threads . . . . . 20
- Virtual Threads Basics . . . . . 20
- Thread-local Variables . . . . . 20
- Scheduling of Virtual Threads . . . . . 20
- Execution of Virtual Threads . . . . . 20
- `ExecutorService` . . . . . 20
- Debugging . . . . . 21

**Structured Concurrency** . . . . . **22**
- Basic Usage . . . . . 22
- `invokeAll` . . . . . 22
- `invokeAny` . . . . . 22
- Shutdown and Close of StructuredTaskScope . . . . . 22

**Scoped Values** . . . . . **23**

# Project Panama . . . . . 24

**Foreign Function & Memory API** . . . . . **25**

Foreign Memory . . . 25
Memory Segments . . . 25
Arenas . . . 25
Dereferencing Memory Segments . . . 25
Memory Layouts . . . 25
Dereferencing Memory Layouts . . . 26
More About Arenas . . . 26
Foreign Functions . . . 26

**Vector API** . . . **28**
API Basics . . . 28
Euclidean Distance Calculation . . . 28

# JVM . . . 29

**Generational ZGC** . . . **30**

**Prepare to Disallow the Dynamic Loading of Agents** . . . **31**

# Deprecated & Removed Features . . . 32

**Deprecated Features** . . . **33**
Finalization . . . 33
Windows 32-bit x86 Port . . . 35

# Miscellaneous . . . 36

**Linux/RISC-V Port** . . . **37**

**Annotation Processing** . . . **38**

# Introduction

Java 21 is released on September 19, 2023. Java 21 is the next LTS version after Java 17. It's expected that users of Java 17 will gradually migrate to Java 21. This book summaries all major changes from Java 17 to Java 21, so you can easily migrate from Java 17 to Java 21.

The most important change in Java 21 is the introduction of virtual threads. Virtual threads will dramatically change multi-threading programming in Java.

> If you are currently using Java 11, check out my book **From Java 11 to Java 17** for upgrading from Java 11 to Java 17 first.

## JEPs in Different Java Releases

Java Enhancement Proposal (JEP) is used to organize changes to OpenJDK. A Java release may contain a list of JEPs. Below are lists of JEPs added in each Java release from Java 18 to Java 21.

### Java 18

Java 18 has 9 JEPs, see the table below.

**Figure 1. JEPs in Java 18**

| Number | Name |
|---|---|
| 400 | UTF-8 by Default |
| 408 | Simple Web Server |
| 413 | Code Snippets in Java API Documentation |
| 416 | Reimplement Core Reflection with Method Handles |
| 417 | Vector API (Third Incubator) |
| 418 | Internet-Address Resolution SPI |
| 419 | Foreign Function & Memory API (Second Incubator) |
| 420 | Pattern Matching for switch (Second Preview) |

| Number | Name |
|---|---|
| 421 | Deprecate Finalization for Removal |

## Java 19

Java 19 has 7 JEPs, see the table below.

**Figure 2. JEPs in Java 19**

| Number | Name |
|---|---|
| 405 | Record Patterns (Preview) |
| 422 | Linux/RISC-V Port |
| 424 | Foreign Function & Memory API (Preview) |
| 425 | Virtual Threads (Preview) |
| 426 | Vector API (Fourth Incubator) |
| 427 | Pattern Matching for switch (Third Preview) |
| 428 | Structured Concurrency (Incubator) |

## Java 20

Java 20 has 7 JEPs, see the table below.

**Figure 3. JEPs in Java 20**

| Number | Name |
|---|---|
| 429 | Scoped Values (Incubator) |
| 432 | Record Patterns (Second Preview) |
| 433 | Pattern Matching for switch (Fourth Preview) |
| 434 | Foreign Function & Memory API (Second Preview) |
| 436 | Virtual Threads (Second Preview) |
| 437 | Structured Concurrency (Second Incubator) |
| 438 | Vector API (Fifth Incubator) |

### Java 21

Java 21 has 15 JEPs, see the table below.

**Figure 4. JEPs in Java 21**

| Number | Name |
|---|---|
| 430 | String Templates (Preview) |
| 431 | Sequenced Collections |
| 439 | Generational ZGC |
| 440 | Record Patterns |
| 441 | Pattern Matching for switch |
| 442 | Foreign Function & Memory API (Third Preview) |
| 443 | Unnamed Patterns and Variables (Preview) |
| 444 | Virtual Threads |
| 445 | Unnamed Classes and Instance Main Methods (Preview) |
| 446 | Scoped Values (Preview) |
| 448 | Vector API (Sixth Incubator) |
| 449 | Deprecate the Windows 32-bit x86 Port for Removal |
| 451 | Prepare to Disallow the Dynamic Loading of Agents |
| 452 | Key Encapsulation Mechanism API |
| 453 | Structured Concurrency (Preview) |

## Install Java 21

You can get Java 21 binaries from several different places. Since Java 21 is a LTS version, many vendors will provide Java 21 releases. For most users, Eclipse Temurin is the best choice.

- Download OpenJDK 21 build from jdk.java.net.
- Download Eclipse Temurin from Adoptium.
- Use SDKMAN to install, e.g. `sdk install java 21.0.2-open` or `sdk install java 21.0.8-tem`.
- Use IDE to download. IntelliJ IDEA can download JDK.

You can also get OpenJDK 21 releases from other vendors, like Amazon Corretto, Oracle, and Zulu. For example, you can use `sdk install java 21.0.8-amzn` to install Amazon Corretto 21.

Below is the output of `java -version` for OpenJDK JDK 21 build.

**Figure 5. Output of java -version**

```
openjdk version "21" 2023-09-19
OpenJDK Runtime Environment (build 21+35-2513)
OpenJDK 64-Bit Server VM (build 21+35-2513, mixed mode, sharing)
```

## Build Tools

When using Java 21 with build tools, Maven or Gradle, you may need to add the `--enable-preview` argument to enable preview features. The following code shows an example of the configuration of Maven compiler plugin. `jdk.incubator.vector` is an incubating module, so it needs to be added explicitly using `--add-modules`.

**Figure 6. Maven compiler plugin configuration**

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.13.0</version>
    <configuration>
        <source>21</source>
        <target>21</target>
        <compilerArgs>
            <arg>--enable-preview</arg>
            <arg>--add-modules</arg>
            <arg>jdk.incubator.vector</arg>
        </compilerArgs>
    </configuration>
</plugin>
```

## Source Code

Source code of this book can be found on GitHub.

# Java Language

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# String Templates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Template Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## String Template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Template Processor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## SMS Message Template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Format Processor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Unnamed Classes and Instance Main Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Java Program Launch Protocol

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Unnamed Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Code Snippets in Java API Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Inline Snippets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## External Snippets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Markup Tags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Regions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Highlight Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Replace Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Link Text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Tag References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Pattern Matching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Record Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Nested Record Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Generic Record Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Pattern Matching for `switch`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### `null` in `switch`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Guarded Case Label

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Enum Constants in `switch`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Unnamed Patterns and Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Unnamed Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Unnamed Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Core Library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# UTF-8 By Default

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Internet-Address Resolution SPI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Simple Web Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Sequenced Collections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `SequencedCollection`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `SequencedSet`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `SequencedMap`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Classes Hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Reimplement Core Reflection with Method Handles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Key Encapsulation Mechanism API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Multi-threading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Create Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Virtual Threads Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Thread-local Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Scheduling of Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Execution of Virtual Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `ExecutorService`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Thread Dump

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### JFR

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Structured Concurrency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Basic Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `invokeAll`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## `invokeAny`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Shutdown and Close of StructuredTaskScope

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Scoped Values

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Project Panama

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Foreign Function & Memory API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Foreign Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Memory Segments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Arenas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Dereferencing Memory Segments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Memory Layouts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Padding Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Sequence Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Group Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Dereferencing Memory Layouts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## More About Arenas

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Foreign Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

### Downcall - Call from Java to Native

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Upcall - Call From Native To Java

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# Vector API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## API Basics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

## Euclidean Distance Calculation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/java17to21.

# JVM

# Generational ZGC

*Generational* ZGC is added in Java 21 (JEP 439).

ZGC was ready for production use since JDK 15 (JEP 377). ZGC is a scalable low-latency garbage collector. Generational ZGC can improve application performance by extending ZGC to maintain separate generations for young and old objects. Young objects will be collected more frequently.

To enable Generational ZGC, the `-XX:+ZGenerational` option should be used together with `-XX:+UseZGC` option. Using only the `-XX:+UseZGC` option will select non-generational ZGC.

**Figure 7. Use Generational ZGC**

```
$ java -XX:+UseZGC -XX:+ZGenerational
```

In Java 23□Generational ZGC become the default option (JEP 474). The non-generational mode is deprecated and will be removed in a future release. Use the `-XX:-ZGenerational` option to select non-generational ZGC.

# Prepare to Disallow the Dynamic Loading of Agents

> *Prepare to Disallow the Dynamic Loading of Agents* is added in Java 21 (JEP 451).

Agents are powerful tools to work with JVM. An agent can alter the code of an application while the application is running. Agents can transform classes during class loading, or redefine classes loaded earlier. JDK 6 introduced the Attach API, which allows a tool to connect to a running JVM and communicate with that JVM. The Attach API also allows loading an agent dynamically into a running JVM.

Dynamically loading an agent can be dangerous, given that an agent can modify arbitrary classes in a running application.

In Java 21, when an agent is loaded dynamically, a warning is issued. To allow agents loaded dynamically without warnings, the `-XX:+EnableDynamicAgentLoading` option must be provided.

The dynamic loading of agents will be disallowed by default in a future Java release. In that case, the `-XX:+EnableDynamicAgentLoading` option will be required.

Agents can still be loaded at startup using the `-javaagent` option, the `-agentlib` option, and the `Launcher-Agent-Class` JAR file attribute.

# Deprecated & Removed Features

Many features have been marked as deprecated or removed since Java 17. You need to pay special attention to these features if your application still uses them.

# Deprecated Features

Below are deprecated features in Java 21.

## Finalization

Finalization is a special mechanism for Java programs to interact with the JVM's garbage collector. Finalization was introduced in JDK 1.0, so it has been there for quite a long time. The original goal of finalization was to provide a safety net to guarantee the release of native resources.

Native resources are managed by the operating system. Typical examples of native resources are file handles and native memory. Native resources are limited and must be carefully managed. It's very important that native resources are not leaked.

Native resources are held by Java objects. These Java objects provide methods to release native resources. For example, a`File` object has the `close()` method to close the file and release the native file handle. The `close()` method must be called to ensure the close of file handles.

Programming errors are inevitable. If developers forgot to call the `close()` method of a `File` object, after this `File` object is claimed by the garbage collector, there will be no reference to this `File` object and it's impossible to release the native file handle. This native file handle is leaked.

The idea of finalization is quite simple. It relies on the interaction with the garbage collector. The garbage collector knows when a object becomes eligible for collection. The garbage collector can run some actions before it reclaims the object.

### `finalize` in `Object`

The method `finalize` of the class `Object` is what Java programmers can use to interact with the garbage collector. The method `finalize` is called by the garbage collector on an object when there are no more references to the object.

If an object needs to release native resources, the release logic can be put into the method `finalize`. Before the object is reclaimed by the garbage collector, native resources can be released by calling the method `finalize`.

Finalization sounds a brilliant idea which can guarantee the release of native resources. Unfortunately, allowing Java programs to interact with the garbage collector has never been a good idea. It turns out that finalization brings more problems than it solves.

Below is a list of common issues with finalization.

- *Unpredictable latency*. During the running of a Java program, it's unpredictable when the garbage collector will run. When an object becomes eligible for reclaim, it's unpredictable when this object will be reclaimed. The delay can be quite long, or even infinite. This means that after a object can be reclaimed and the native resource held by this object can be safely released, the native resource may be actually released after a long delay. Even worse, the finalizer may be never run by the garbage collector.
- *Unconstrained behavior*. You can put any code in the `finalize` method. This may cause unexpected behavior when finalizer runs. Finalizer is called when this object becomes unreachable and ready for reclaim. If a reference is saved to the object being finalized, this object will be resurrected and becomes reachable again.
- *Always enabled*. When finalizer is added to a class, it's enabled for all instances of this class. Finalization cannot be cancelled, or disabled for a single object.

After finalization is deprecated, we should use different ways to release native resources. The simplest choice is to release the resource right away when it's no longer used. For example, if we finish the use of a `File` object, we can call its method `close()` to release it. This is usually done with `try-with-resources`. This requires that the object's class must implement the interface `java.lang.AutoClosable`. The method `close()` contains the logic to release native resource. `try-with-resources` makes sure that method `close()` will always be called.

The limitation of `try-with-resources` is that the native resource will be released after statements in `try` block finish execution, either normally or exceptionally. This means that the resource cannot be held for a long time.

**Figure 8. Usage of try-with-resources**

```
public class TryWithResources {

  void copyFile() throws IOException {
    try (var input = new FileInputStream("input.txt");
         var output = new FileOutputStream("output.txt")) {
      input.transferTo(output);
    }
  }
}
```

## Windows 32-bit x86 Port

Windows 32-bit x86 port of OpenJDK was deprecated for removal in Java 21. The main reason for this deprecation is that Windows 10, the last Windows operating system to support 32-bit operation, will reach End of Life in October 2025. So it's not necessary to keep maintaining this port.

Another issue with Windows 32-bit x86 port is that the implementation of virtual threads for Windows x86-32 falls back to the use of kernel threads.

After this deprecation, attempting to configure a Windows x86-32 build will produce an error. The new build configuration option `--enable-deprecated-ports=yes` should be used to suppress the error and continue.

> Windows 32-bit x86 port of OpenJDK was finally removed in Java 24 (JEP 479).

# Miscellaneous

This part covers miscellaneous changes in Java 21.

# Linux/RISC-V Port

*Linux/RISC-V Port* is added in Java 19 (JEP 422).

JDK is ported to Linux/RISC-V.

RISC-V port can be downloaded from here, see the screenshot below.



| | | |
|---|---|---|
| openjdk-jdk21-linux-riscv64-server-fastdebug-gcc10-glibc2.31.tar.xz | 187.4 MiB | 2023-Nov-03 01:26 |
| openjdk-jdk21-linux-riscv64-server-fastdebug-gcc12-glibc2.36.tar.xz | 186.8 MiB | 2023-Nov-02 22:13 |
| openjdk-jdk21-linux-riscv64-server-fastdebug-gcc8-glibc2.28.tar.xz | 187.0 MiB | 2023-Nov-03 00:52 |
| openjdk-jdk21-linux-riscv64-server-optimized-gcc10-glibc2.31.tar.xz | 163.1 MiB | 2023-Nov-03 01:40 |
| openjdk-jdk21-linux-riscv64-server-optimized-gcc12-glibc2.36.tar.xz | 163.6 MiB | 2023-Nov-02 22:28 |
| openjdk-jdk21-linux-riscv64-server-optimized-gcc8-glibc2.28.tar.xz | 166.2 MiB | 2023-Nov-03 01:06 |
| openjdk-jdk21-linux-riscv64-server-release-gcc10-glibc2.31.tar.xz | 162.1 MiB | 2023-Nov-03 01:17 |
| openjdk-jdk21-linux-riscv64-server-release-gcc12-glibc2.36.tar.xz | 162.6 MiB | 2023-Nov-02 22:02 |
| openjdk-jdk21-linux-riscv64-server-release-gcc8-glibc2.28.tar.xz | 165.4 MiB | 2023-Nov-03 00:44 |
| openjdk-jdk21-linux-riscv64-server-slowdebug-gcc10-glibc2.31.tar.xz | 135.6 MiB | 2023-Nov-03 01:33 |
| openjdk-jdk21-linux-riscv64-server-slowdebug-gcc12-glibc2.36.tar.xz | 136.5 MiB | 2023-Nov-02 22:20 |
| openjdk-jdk21-linux-riscv64-server-slowdebug-gcc8-glibc2.28.tar.xz | 135.4 MiB | 2023-Nov-03 00:59 |

**Figure 9. RISC-V port download**

# Annotation Processing

Starting from JDK 6, `javac` scans the source file for annotations and then the class path for matching annotation processors. This is the default behavior in JDK <= 22.

How `javac` processes annotations is controlled by the option `-proc:$policy`, where `$policy` can have the following values.

- `none`: compilation without annotation processing
- `only`: annotation processing without compilation
- `full`: annotation processing followed by compilation

`-proc:full` is added in JDK 21. `-proc:none` is the default on JDK 23. In JDK 21, `javac` prints an informative message if implicit usage of annotation processing under the default policy is detected.

If your application relies on `javac`'s default annotation processing behavior, you should add the `-proc:full` option to `javac`, then your application won't be affected after upgrading to JDK 23.