

Performance & Security Improvements



Sander Mak

Software developer & architect

@sander_mak





Virtual Threads in Detail:

What's New in Java 21

Sander Mak



Virtual Threads

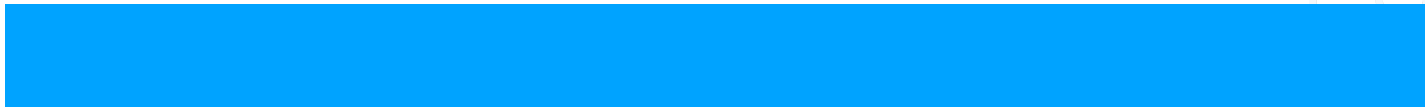
Virtual Thread 1



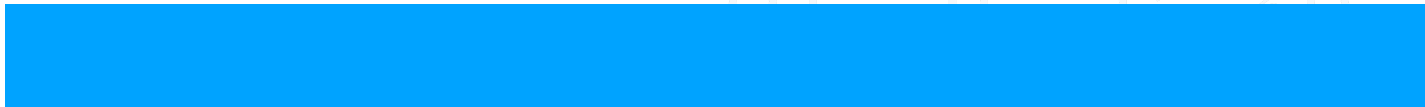
Virtual Thread 2



Platform Thread 1



Platform Thread 2



Virtual Threads: Pinning

Virtual Thread 1

Running

Synchronized block/method

Virtual Thread 2

Running

```
public synchronized void heavyWork() {  
    // long-running, blocking operations  
}
```

Platform Thread 1

Synchronized block/method, holding lock



Virtual Threads: Synchronize without Pinning

As of Java 24, no more pinning within synchronized blocks

Replacing `synchronized` with
`java.util.concurrent.locks.*` no longer required



Virtual threads can still be pinned through native code execution



Security Improvements

`sun.misc.Unsafe`

Direct memory access methods

Deprecated in Java 23

Log run-time warning in Java 24

`VarHandle` (JDK 9)

Foreign Function & Memory API (JDK 22)

Java Native Integration (JNI)

Opt-in for unsafe functionality
in future, warning for now

`--enable-native-access`

`--illegal-native-access`



Security Improvements: Quantum Resistant Cryptography

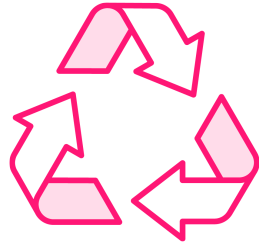
Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM)

Module-Lattice-Based Digital Signature Algorithm (ML-DSA)

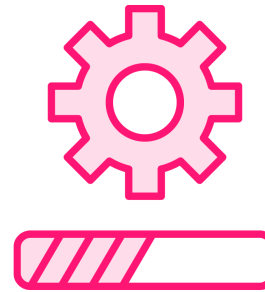
NIST & FIPS compliant



Performance Improvements



Garbage Collection

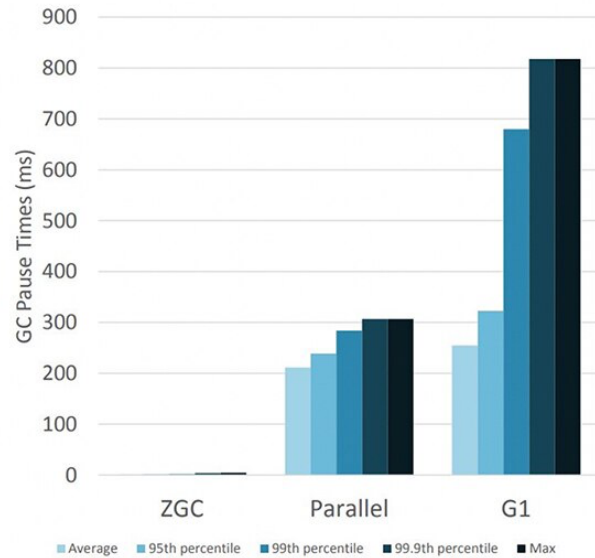


Ahead-of-Time (AOT)
Class Loading and Linking

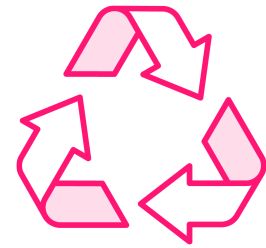
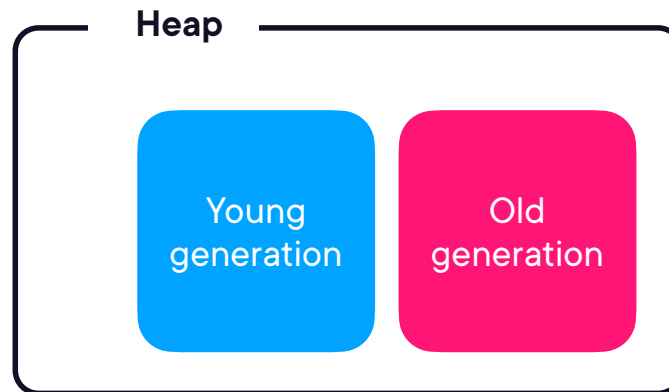


Generational ZGC

-XX:+UseZGC

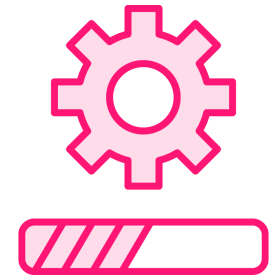


bit.ly/zgc-benchmark



AOT Class Loading and Linking

Decreasing JVM start-up time by making application classes instantly available, creating an ahead-of time cache containing their loaded and linked representations.



AOT Class Loading and Linking

Training run for AOT cache

```
java -XX:AOTMode=record \  
-XX:AOTConfiguration=myapp.aotconf \  
-cp myapp.jar com.ps.MyApp
```

1

Create AOT
cache file

```
java -XX:AOTMode=create \  
-XX:AOTConfiguration=myapp.aotconf \  
-XX:AOTCache=myapp.aot \  
-cp myapp.jar
```

2

Run application with
pre-loaded & linked classes

```
java -XX:AOTCache=myapp.aot \  
-cp myapp.jar com.ps.MyApp
```

3

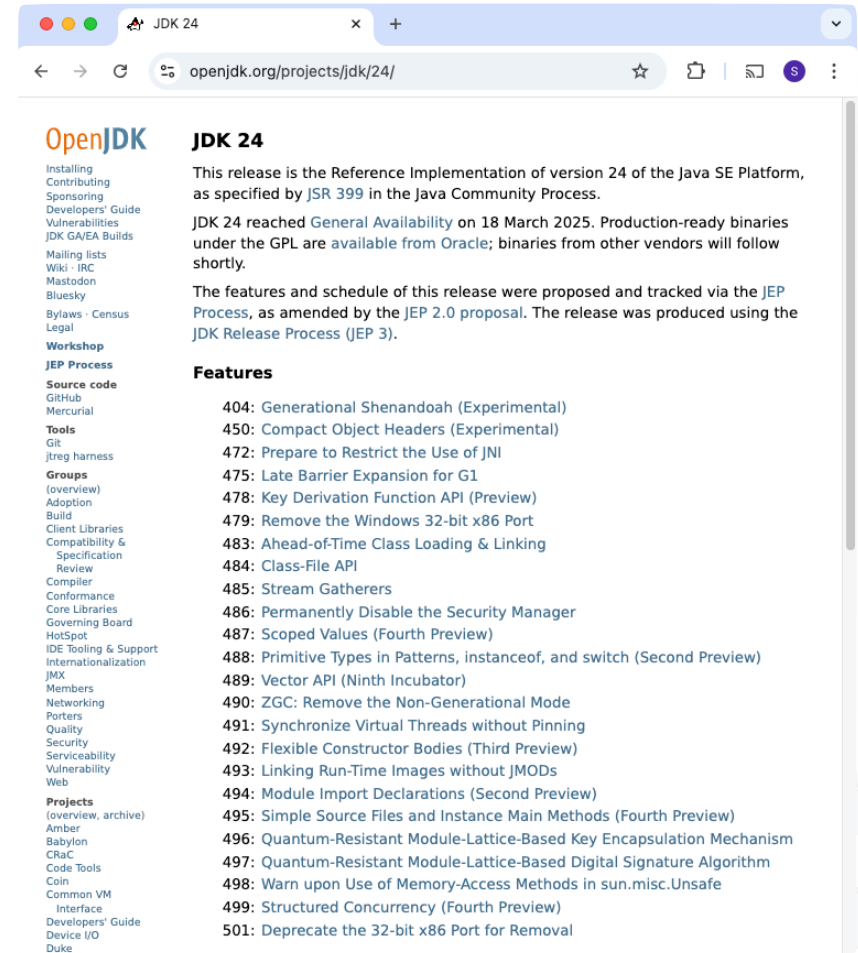
~40% decrease in start-up time observed



Course Wrap-up

JEPs for all major features

<https://openjdk.org/projects/jdk/24/>



The screenshot shows the OpenJDK 24 project page in a web browser. The browser's address bar displays the URL `openjdk.org/projects/jdk/24/`. The page features a sidebar on the left with a navigation menu containing links such as 'Installing', 'Contributing', 'Sponsoring', 'Developers' Guide', 'Vulnerabilities', 'JDK GA/EA Builds', 'Mailing lists', 'Wiki - IRC', 'Mastodon', 'Bluesky', 'Bylaws - Census', 'Legal', 'Workshop', 'JEP Process', 'Source code', 'GitHub', 'Mercurial', 'Tools', 'Git', 'Jtreg harness', 'Groups', 'Adoption', 'Build', 'Client Libraries', 'Compatibility & Specification', 'Review', 'Compiler', 'Conformance', 'Core Libraries', 'Governing Board', 'HotSpot', 'IDE Tooling & Support', 'Internationalization', 'JMX', 'Members', 'Networking', 'Porters', 'Quality', 'Security', 'Serviceability', 'Vulnerability', 'Web', 'Projects', 'Amber', 'Babyon', 'CRAc', 'Code Tools', 'Coin', 'Common VM', 'Interface', 'Developers' Guide', 'Device I/O', and 'Duke'. The main content area is titled 'JDK 24' and contains the following text: 'This release is the Reference Implementation of version 24 of the Java SE Platform, as specified by JSR 399 in the Java Community Process.' It also states: 'JDK 24 reached General Availability on 18 March 2025. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.' A paragraph follows: 'The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal. The release was produced using the JDK Release Process (JEP 3).' Below this, a section titled 'Features' lists 20 JEPs, each with a number and a description, such as '404: Generational Shenandoah (Experimental)' and '501: Deprecate the 32-bit x86 Port for Removal'.

OpenJDK

- Installing
- Contributing
- Sponsoring
- Developers' Guide
- Vulnerabilities
- JDK GA/EA Builds
- Mailing lists
- Wiki - IRC
- Mastodon
- Bluesky
- Bylaws - Census
- Legal
- Workshop
- JEP Process
- Source code
- GitHub
- Mercurial
- Tools
- Git
- Jtreg harness
- Groups
- (overview)
- Adoption
- Build
- Client Libraries
- Compatibility & Specification
- Review
- Compiler
- Conformance
- Core Libraries
- Governing Board
- HotSpot
- IDE Tooling & Support
- Internationalization
- JMX
- Members
- Networking
- Porters
- Quality
- Security
- Serviceability
- Vulnerability
- Web
- Projects
- (overview, archive)
- Amber
- Babyon
- CRAc
- Code Tools
- Coin
- Common VM
- Interface
- Developers' Guide
- Device I/O
- Duke

JDK 24

This release is the Reference Implementation of version 24 of the Java SE Platform, as specified by JSR 399 in the Java Community Process.

JDK 24 reached General Availability on 18 March 2025. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal. The release was produced using the JDK Release Process (JEP 3).

Features

- 404: Generational Shenandoah (Experimental)
- 450: Compact Object Headers (Experimental)
- 472: Prepare to Restrict the Use of JNI
- 475: Late Barrier Expansion for G1
- 478: Key Derivation Function API (Preview)
- 479: Remove the Windows 32-bit x86 Port
- 483: Ahead-of-Time Class Loading & Linking
- 484: Class-File API
- 485: Stream Gatherers
- 486: Permanently Disable the Security Manager
- 487: Scoped Values (Fourth Preview)
- 488: Primitive Types in Patterns, instanceof, and switch (Second Preview)
- 489: Vector API (Ninth Incubator)
- 490: ZGC: Remove the Non-Generational Mode
- 491: Synchronize Virtual Threads without Pinning
- 492: Flexible Constructor Bodies (Third Preview)
- 493: Linking Run-Time Images without JMODs
- 494: Module Import Declarations (Second Preview)
- 495: Simple Source Files and Instance Main Methods (Fourth Preview)
- 496: Quantum-Resistant Module-Lattice-Based Key Encapsulation Mechanism
- 497: Quantum-Resistant Module-Lattice-Based Digital Signature Algorithm
- 498: Warn upon Use of Memory-Access Methods in sun.misc.Unsafe
- 499: Structured Concurrency (Fourth Preview)
- 501: Deprecate the 32-bit x86 Port for Removal

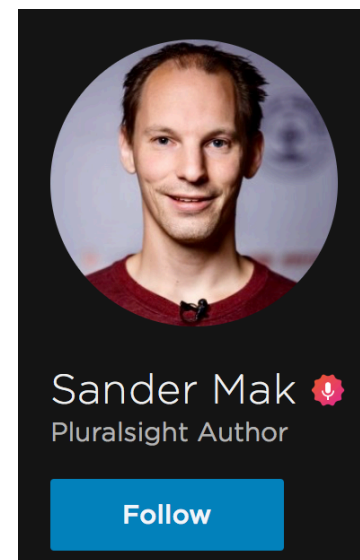


More Information

What's New in Java 9-21

Sander Mak

Follow for course updates



bit.ly/ps-sander

