Java 18 & 19

2022 State of the Java Ecosystem Report

- Java 11 is the new standard
  - Java 11 2020 11.11% → 2022 48.44%
  - Java 8 2020 84.48% → 2022 46.45%
- Source
  - https://newrelic.com/resources/report/2022-state-of-java-ecosystem

Oracle Java SE Support Roadmap

Oracle Java SE Support Roadmap*†				
Release	**GA Date**	**Premier Support Until**	**Extended Support Until**	**Sustaining Support**
7 (LTS)	July 2011	July 2019	July 2022*****	Indefinite
8 (LTS)**	March 2014	March 2022	December 2030*****	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	September 2026	Indefinite
12 (non-LTS)	March 2019	September 2019	Not Available	Indefinite
13 (non-LTS)	September 2019	March 2020	Not Available	Indefinite
14 (non-LTS)	March 2020	September 2020	Not Available	Indefinite
15 (non-LTS)	September 2020	March 2021	Not Available	Indefinite
16 (non-LTS)	March 2021	September 2021	Not Available	Indefinite
17 (LTS)	September 2021	September 2026****	September 2029****	Indefinite
18 (non-LTS)	March 2022	September 2022	Not Available	Indefinite
19 (non-LTS)***	September 2022	March 2023	Not Available	Indefinite
20 (non-LTS)***	March 2023	September 2023	Not Available	Indefinite
21 (LTS)***	September 2023	September 2028	September 2031	Indefinite

JDK 18

- 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)
- 421: Deprecate Finalization for Removal

JDK 19

- 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)

https://openjdk.org/

Incubator vs Preview

- Incubator(孵化中) 功能需 import incubator module 才能使用。
- Preview 功能需要 Compile 時要帶上— enable-Preview 才能使用 (Maven 專案可於 pom.xml 設定)
- 兩者都不建議於 Production 環境使用。
- Preview 功能已經接近完成，但還需要一些 feeback ，確保最後的結果符合需求； Incubator 功能可能還會有幾次的迭代調整，但仍希望能先收到一些 developer feeback(https://t.co/35T2O1W9kN) 。

JEP 400: UTF-8 by Default

- In JDK 17 and earlier, the default charset is determined when the Java runtime starts.
- Make Java programs more predictable and portable when their code relies on the default charset.
- Several standard Java APIs use the default charset, including:
  - java.io package, InputStreamReader, FileReader, OutputStreamWriter, FileWriter, and PrintStream.
  - java.util package, Formatter and Scanner.
  - java.net package, URLEncoder and URLDecoder.

JEP 408: Simple Web Server

- Provide a command-line tool to start a minimal web server that serves static files only.
- Provide a default implementation via the command line together with a small API for programmatic creation and customization.
- Only idempotent HEAD and GET requests are served. Any other requests receive a 501 - Not Implemented or a 405 - Not Allowed response.
- The Simple Web Server supports HTTP/1.1 only. There is no HTTPS support.

JEP 413: Code Snippets in Java API Documentation

- Introduce an @snippet tag for JavaDoc's Standard Doclet, to simplify the inclusion of example source code in API documentation.
- Facilitate the validation of source code fragments, by providing API access to those fragments. Although correctness is ultimately the responsibility of the author, enhanced support in javadoc and related tools can make it easier to achieve.
- Enable modern styling, such as syntax highlighting, as well as the automatic linkage of names to declarations.
- Enable better IDE support for creating and editing snippets.

Pattern Matching for switch

- JEP 420 (JDK 18 Second Preview), JEP 427 (JDK 19 Third Preview)
- Expand the expressiveness and applicability of switch expressions and statements by allowing patterns to appear in case labels.
- Allow the historical null-hostility of switch to be relaxed when desired.
- Increase the safety of switch statements by requiring that pattern switch statements cover all possible input values.

JEP 421: Deprecate Finalization for Removal

- finalization has several critical, fundamental flaws:
  - Unpredictable latency — An arbitrarily long time may pass between the moment an object becomes unreachable and the moment its finalizer is called. In fact, the GC provides no guarantee that any finalizer will ever be called.
  - Unconstrained behavior — Finalizer code can take any action. In particular, it can save a reference to the object being finalized, thereby resurrecting the object and making it reachable once again.
  - Always enabled — Finalization has no explicit registration mechanism. A class with a finalizer enables finalization for every instance of the class, whether needed or not. Finalization of an object cannot be cancelled, even if it is no longer necessary for that object.
  - Unspecified threading — Finalizers run on unspecified threads, in an arbitrary order. Neither threading nor ordering can be controlled.
- Maintainers of libraries and applications that rely upon finalization should consider migrating to other resource management techniques such as the try-with-resources statement and cleaners.

JEP 405: Record Patterns (Preview)

- Extend pattern matching to express more sophisticated, composable data queries.
- Do not change the syntax or semantics of type patterns.

JEP 425: Virtual Threads (Preview)

- Enable server applications written in the simple thread-per-request style to scale with near-optimal hardware utilization.
- Enable existing code that uses the java.lang.Thread API to adopt virtual threads with minimal change.
- Enable easy troubleshooting, debugging, and profiling of virtual threads with existing JDK tools.

JEP 425: Virtual Threads (Preview)

- virtual threads can significantly improve application throughput when
  - The number of concurrent tasks is high (more than a few thousand), and
  - The workload is not CPU-bound, since having many more threads than processor cores cannot improve throughput in that case.

JEP 425: Virtual Threads (Preview)

- Do not pool virtual threads
  - virtual threads are not expensive and there is never a need to pool them

JEP 428: Structured Concurrency (Incubator)

- Improve the maintainability, reliability, and observability of multithreaded code.
- Promote a style of concurrent programming which can eliminate common risks arising from cancellation and shutdown, such as thread leaks and cancellation delays.

Others

- 416: Reimplement Core Reflection with Method Handles
- 417: Vector API (Third Incubator)
- 418: Internet-Address Resolution SPI
- 419: Foreign Function & Memory API (Second Incubator)
- 422: Linux/RISC-V Port

End