Le versioni di Java dalla 8 in poi

🚀 Java 8 (2014)

- **Lambda Expressions** → permettono di scrivere funzioni inline (`(x) -> x * 2`).
- **Functional Interfaces** → interfacce con un solo metodo astratto (`@FunctionalInterface`).
- **Stream API** → operazioni su collezioni in stile funzionale (`map`, `filter`, `reduce`).
- **Optional** → wrapper per gestire valori null in modo più sicuro.
- **Date and Time API (java.time)** → nuova gestione di date e orari (ispirata a Joda-Time).
- **Default e static methods nelle interfacce**.

🚀 Java 9 (2017)

- **Java Platform Module System (JPMS)** → suddivisione del codice in moduli.
- **JShell (REPL)** → esecuzione interattiva di codice Java da console.
- **Miglioramenti alle API Stream** (`takeWhile`, `dropWhile`, `iterate`).
- **Factory methods per collezioni immutabili** (`List.of(...)`, `Set.of(...)`).
- **Interfacce private methods**.

🚀 Java 10 (2018)

- **Local-variable type inference** (`var`) → inferenza automatica del tipo.
- **Garbage Collector improvements** → introduzione di G1 come default.
- **CopyOf per collezioni immutabili**.

🚀 Java 11 (2018 LTS)

- **String API improvements** (`isBlank`, `lines`, `strip`).
- **HttpClient API** (standard, non più incubator).
- **Esecuzione di file `.java` senza compilazione** (`java Hello.java`).
- **Nuovi metodi in `Files` (readString, writeString)`**.
- **Deprecazione di Java EE e CORBA**.

🚀 Java 12 (2019)

- **Switch Expressions (preview)** → `switch` che ritorna valori.
- **Shenandoah GC e ZGC (nuovi Garbage Collector sperimentali)**.
- **Compact Number Formatting**.

🚀 Java 13 (2019)

- **Text Blocks (preview)** → stringhe multilinea con `"""`.
- **Switch Expressions migliorati**.
- **Dynamic CDS (Class Data Sharing)**.

🚀 Java 14 (2020)

- **Records (preview)** → classi compatte per DTO (`record Point(int x, int y)`).
- **Pattern Matching per `instanceof` (preview)**.
- **Helpful NullPointerExceptions** → messaggi più chiari.

🚀 Java 15 (2020)

- **Sealed Classes (preview)** → restrizione delle classi che possono estendere una superclasse.
- **Text Blocks** diventano definitivi.
- **Hidden Classes** (supporto per strumenti e framework dinamici).

🚀 Java 16 (2021)

- **Records definitivi**.
- **Pattern Matching per `instanceof` definitivo**.
- `Stream.toList()` metodo diretto.
- **Vector API (incubator)**.

🚀 Java 17 (2021 LTS)

- **Sealed Classes definitive**.
- **Pattern Matching per `switch` (preview)**.
- **Nuove API per Random Number Generators**.
- **Rimozione di vecchie librerie (Applet, RMI Activation)**.
- **JDK come una singola immagine (no più JRE separato)**.

🚀 Java 18 (2022)

- **Simple Web Server** (`jwebserver`).
- **UTF-8 come charset di default**.
- **Vector API migliorato**.

🚀 Java 19 (2022)

- **Virtual Threads (preview, Project Loom).**
- **Pattern Matching per `switch` avanzato (preview).**
- **Structured Concurrency (incubator).**

🚀 Java 20 (2023)

- **Record Patterns (preview)**.
- **Virtual Threads migliorati**.
- **Scoped Values (incubator)**.

🚀 Java 21 (2023 LTS)

- **Virtual Threads definitivi** 🎉 (Project Loom).
- **Pattern Matching per `switch` definitivo**.
- **Sequenced Collections** (`SequencedSet`, `SequencedMap`).
- **String Templates (preview)** → interpolazione di stringhe.
- **Key Encapsulation Mechanism API (cripto)**.

🚀 Java 22 (2024)

- **Statements `switch` migliorati (preview)** → semplificazione ulteriore.
- **String Templates (seconda preview)** → più potenti e raffinati.
- **Unnamed Variables e Patterns (preview)** → introduzione di `_` come placeholder.
- **Foreign Function & Memory API (third preview)** → interfacciamento con codice nativo senza JNI.
- **Scoped Values (second incubator)** → condivisione sicura di dati tra Virtual Threads.

🚀 Java 23 (2024)

- **String Templates (terza preview)** → verso la stabilizzazione.
- **Primitive Types in Patterns (preview)** → supporto migliorato per `switch` e matching sui primitivi.
- **Stream Gatherers (preview)** → nuovo modo di trasformare stream (es. sliding window).
- **Foreign Function & Memory API (final)** 🎉 → API stabile per interfacciarsi con memoria e librerie native.
- **Structured Concurrency (second preview)** → gestione concorrente dei task semplificata.

🚀 Java 24 (2025)

- Stabilizzazione di **String Templates**.
- Miglioramenti a **Structured Concurrency**.
- Evoluzioni su **Pattern Matching** e sulle **Sequenced Collections**.
- Possibili nuove API per sicurezza e criptografia.