Java version and their Features

Java 8 (March 2014)

Key Features:

- **Lambda Expressions** for concise, functional-style coding.
- **Functional Interfaces** (annotated with @FunctionalInterface).
- **Default Methods** in interfaces—allow method implementations.
- **Stream API** for declarative collection processing.(Aegis Softtech)

Advantages:

- Greatly improves code brevity and readability via lambdas and streams.
- Encourages functional programming patterns, enabling clean and parallel processing.

Disadvantages:

- Can introduce complexity—especially for beginners unaware of functional styles.
- Streams may produce performance overhead in some use cases.(arXiv)

Java 9 (September 2017)

Key Features:

- **Java Platform Module System (JPMS/Jigsaw)** for modularization and encapsulation.(Baeldung on Kotlin, Wikipedia)
- **JShell (REPL)** for interactive coding.(DigitalOcean, GeeksforGeeks)
- **Factory methods** like List.of(), for immutable collections.(Medium, GeeksforGeeks)
- Enhancements in Stream API (takeWhile, dropWhile, ofNullable).(GeeksforGeeks)
- **Private methods in interfaces**, improved try-with-resources, HTTP/2 client, improved documentation (Javadoc), Process API, and more.(DigitalOcean, GeeksforGeeks)

Advantages:

- Better modularity and encapsulation thanks to JPMS.
- JShell makes experimentation and learning much easier.
- Factory methods and Stream improvements simplify common tasks.

Disadvantages:

- JPMS can be complex to configure, particularly for legacy systems.
- Developers had to refactor existing builds and fix modularization issues.

Java 10 (March 2018)

Key Features:

- **Local-Variable Type Inference** (var) for cleaner declarations.(JavaTechOnline)
- New Collectors.unmodifiableList(), and copyOf() methods for immutable collections.(JavaTechOnline)
- Optional.orElseThrow() convenience method.(JavaTechOnline)
- Garbage collector enhancements.(JavaTechOnline)

Advantages:

- Makes code more concise without losing type safety.
- Stronger immutable collection support.

Disadvantages:

- Overuse of var can make code less readable (if not used judiciously).
- Short support cycle—only six months.

Java 11 (September 2018) — LTS

Key Features:

- **Collection.toArray(IntFunction)** for type-safe array creation.(JavaTechOnline)
- **var in lambda parameters** for consistency across contexts.(JavaTechOnline)
- Introduced license changes—now free for personal use but restricted for commercial production.(JavaTechOnline)

Advantages:

- Recognized as a stable, long-term support release.
- Further improves syntax convenience and type handling.

Disadvantages:

- License changes may require enterprises to secure subscriptions for production use.

Java 12–14 (2019–2020)

Key Features:

- **Switch Expressions** (preview in Java 12–13, standard in 14) offering concise, expression-based pattern switching.(Advanced Web, JavaTechOnline)
- **Helpful NullPointerExceptions**, more specific messaging (preview then standard).(Advanced Web)
- **Text Blocks** for multi-line string literals (preview in 13, standard in 14–15).(Advanced Web)

Advantages:

- Cleaner, safer, more expressive control structures and strings.
- Better debugging for null pointer issues.

Disadvantages:

- Initially available as previews, requiring explicit flags to use.
- Early adoption sometimes unstable or unclear.

Java 15–17 (2020–2021)

Key Features:

- **Text Blocks** finalized.(Advanced Web)
- **Record Classes**—compact syntax for immutable data carriers (preview in 14–15; standard in 16).(Advanced Web)
- **Pattern Matching for instanceof** to simplify type checks (preview then standard).(Advanced Web)
- **Sealed Classes** to restrict subclassing (preview then standard).(Advanced Web)

Advantages:

- Records reduce boilerplate and improve immutability.
- Pattern matching and sealed classes make code safer and more maintainable.
- Java 17 is LTS—stable and widely adopted.

Disadvantages:

- Preview features again involved complexity in enabling and adoption.
- Learning curve for new type system improvements.

Java 18–21 (2022–2023) — including LTS at 21

Key Features:

- **Pattern matching for switch** adds power and expressiveness.(Advanced Web, JetBrains)
- **Record Patterns**, **Unnamed Variables**, **String Templates**, **Unnamed Classes and Instance Main Methods** introduced (mostly preview in 21).(Advanced Web, JetBrains)

Advantages:

- Offers a highly expressive and terse syntax style.
- Easier data destructuring and cleaner string handling.

Disadvantages:

- Preview nature; not finalized in earlier releases.
- Could cause fragmentation as features evolve between versions.

Beyond — Java 22+ and Future (2024–2025)

Key Notes:

- Java continues with six-month release cadence. Latest official at mid-2025 is Java **24**, with features like **stream gatherers**, flexible constructor bodies, and more.(JetBrains, Marco Behler)
- Ongoing **Project Valhalla** work on value types, inline classes, and generics enhancements.(Wikipedia)

Summary Table: Java Versions & Highlights

Version	Key Highlights	LTS?	Advantages	Caveats
Java 8	Lambdas, Streams, Default Methods	Yes	Modern syntax, functional programming	Learning curve, performance nuances

Java 9	Modules, JShell, Immutable Collections	No	Modularity, interactive REPL	Complexity, migration effort
Java 10	var, immutable collection methods	No	Cleaner declarations	Readability risk with var misuse
Java 11	LTS, array improvements, var in lambdas	Yes	Enterprise-ready, improvements continue	License limitations for production
Java 12–14	Switch expressions, Text Blocks, NPE msgs	No	Cleaner syntax, better debugging	Initial instability in previews
Java 15–17	Records, Pattern matching, Sealed types	Yes	Reduced boilerplate, safer code	Need to learn new paradigms
Java 18–21	Advanced patterns, string templates	Yes (21)	Expressive syntax, future-ready	Many preview features, evolving semantics
Java 22+	Stream gatherers, Project Valhalla	No	Experimental features, better performance	Highly experimental — not production-ready