**Historical Java Versions**

1. **Java 1.0 (1996)**
   * The original release, establishing Java's syntax and introducing key features like the AWT (Abstract Window Toolkit) and Java Applets.
2. **Java 1.1 (1997)**
   * Added inner classes, JavaBeans, JDBC (Java Database Connectivity), and RMI (Remote Method Invocation).
3. **Java 1.2 - Java 1.4 (1998 - 2002)**
   * These versions introduced the Swing library for GUI development, collections framework, and improvements to Java I/O and security.

**Java 5 (2004)**

* Major overhaul introducing generics, annotations, enumerated types, varargs, and the for-each loop.
* Added the java.util.concurrent package, making multi-threading easier.

**Java 6 (2006)**

* Performance improvements, web services support, and enhancements to the scripting API.
* Included support for compiling and interpreting Java code through the javax.script package.

**Java 7 (2011)**

* Language updates like the try-with-resources statement, switch with strings, and binary literals.
* Added NIO.2 (New I/O), which provided a more flexible and powerful file I/O library.

**Java 8 (2014)**

* Landmark version with Lambda expressions, Stream API, and Optional class.
* Introduced the new Date and Time API (java.time package) to replace the older java.util.Date.

**Java 9 (2017)**

* **Module System (Project Jigsaw)**: Allows splitting the JDK into modules, improving scalability and maintainability.
* JShell (interactive REPL for Java), and enhancements to the Stream API.

**Java 10 (2018)**

* Introduction of var for local variable type inference.
* Improved garbage collection (GC) and performance enhancements.

**Java 11 (2018) (LTS)**

* Long-Term Support (LTS) release with features like new String methods, HTTP Client API, and removal of Java Applets.
* var for lambda expressions and simplified JDK runtime distribution.

**Java 12 - Java 13 (2019)**

* Added switch expressions (preview feature), text blocks (preview in 13), and constant dynamic support in JVM.
* Java 13 introduced yield statements within switch expressions.

**Java 14 - Java 15 (2020)**

* **Java 14**: Added records (preview), pattern matching for instanceof, and the Helpful NullPointerExceptions.
* **Java 15**: Sealed classes (preview), and text blocks became a standard feature.

**Java 16 - Java 17 (2021)**

* **Java 16**: Records and pattern matching finalized, along with improvements to the JDK internals.
* **Java 17**: LTS release with finalized sealed classes, new macOS rendering pipeline, and language enhancements.

**Java 18 - Java 20 (2022 - 2023)**

* **Java 18**: Simple Web Server added for testing and development, UTF-8 as the default charset.
* **Java 19**: Virtual threads (preview), structured concurrency for easier multithreading.
* **Java 20**: Pattern matching improvements, foreign function & memory API as previews.

**Java 21 (2023) (LTS)**

* Introduced finalized features like pattern matching for switch, unnamed patterns and variables, and sequenced collections.
* Scoped values and more structured concurrency patterns.
* Improved the Foreign Function & Memory API, advancing interoperability with non-Java code.

**Release Cycle and LTS Versions**

Since Java 9, Oracle adopted a **6-month release cycle**, introducing new features regularly and providing developers with more frequent updates. **LTS (Long-Term Support)** versions (like Java 8, 11, 17, and 21) are supported for extended periods, while non-LTS versions receive shorter-term support.

The LTS versions are recommended for production environments, while non-LTS releases allow developers to experiment with new language and JVM features.