NEW FEATURES BETWEEN JAVA 8 AND JAVA 19

8 October, 2022

Since version 9, Java has new features every 6 months and it's very hard to keep track of these new changes. Most of the information on the internet describes changes between the last 2 Java versions. However, if you're in a similar situation as me, you're not using one of the latest Java versions but a version several releases older.

Then it's useful to know which new features were added since the version you use now, or between the versions that you use now and the one you want to start using next. Therefore I've compiled a table with all new features added since Java 8 for each new Java version and I keep it updated for every newer Java version.

Below you can find tables for:

- New Java language features – additions to the Java language or APIs
- New Java language features in preview/incubation – additions to the Java language or APIs which aren't in their final state yet
- New tools and features in OpenJDK – additions outside of the language, such as command line tools or JVM improvements
- Deprecated/removed features and APIs

> Note that I didn't include all the new features and API additions, only those that are useful for a wide range of developers, to keep the list brief.

This article isn't about choosing the right Java version for you to use. But I'll at least summarize all the Java versions since Java 8 and for your convenience:

- Java 8 – LTS release (Long Term Support), last LTS release before JPMS (modules) were introduced in Java 9
- Java 11 – LTS release
- Java 17 – LTS release, the latest LTS release until September 2023
- Java 19 – non-LTS release, the latest version released in September 2022

- all other Java versions – non-LTS releases

In most cases, you should only use LTS releases in production. That is unless you have specific needs to use the greatest and latest and are willing to upgrade to a new Java version every 6 months. To find out more about Java LTS releases, you can read The art of long-term support and what LTS means for the Java ecosystem by Oracle.

Java language features and APIs

Feature/API	Since	Preview since
Simple Web Server (a.k.a com.sun.net.httpserver.SimpleFileServer) `SimpleFileServer.createFileServer(new InetSocketAddress(9000), path, logLevel).start();` Cmd Line: `jwebserver -p 9000`	18	
Code Snippets in Java API Documentation `{@snippet : lines of code}` instead of `<pre>{@code lines of code }</pre>`	18	
UTF-8 by Default – instead of the charset defined by the system. Run `javac -encoding UTF-8` with JDK 8-17 to check for Java source encoding issues.	18	
Context-Specific Deserialization Filters – allow applications to configure deserialization filters, either specify a pattern via a system property, or a filter factory via a system property or via `ObjectInputFilter.Config`	17	
Sealed classes (inheritance only for allowed classes): `public abstract sealed class Shape permits Circle, Rectangle, Square {...}`	17	15
Record type – data classes with implicit getters, constructor, `equals`, `hashCode` and `toString` methods: `record Point(int x, int y) { }`	16	14

Static members in inner classes (part of Record type JEP)
`new Object() {`
`static record MyData(String data) {};`
`public static final int CONSTANT = 1;`
`};`
	16	16

Pattern Matching for `instanceof`: `if (x instanceOf String s) { String a = s; }`	16	14

Text blocks:
`String query = """`
`SELECT "EMP_ID", "LAST_NAME"`
`FROM "EMPLOYEE_TB";`
`"""`
	15	13

New String methods (`formatted`, `stripIndent`, `translateEscapes`)	15	

Switch expressions:
`boolean isWeekend = switch (day) {`
`case SATURDAY, SUNDAY -> true;`
`default -> false;`
`};`
	14	12

New String methods (`indent`, `transform`)	12	
CompactNumberFormat class	12	
New String methods (`repeat`, `isBlank`, `strip`, `lines`)	11	

`var` type allowed in Lambda Parameters: `(@NonNull var x) -> process(x)`	11	
New HTTP client API	11	9
TLS v1.3 – support for a new SSL/TLS protocol version	11	
`var` type allowed for local variables: `var length = str.length();`	10	
Flow API (reactive streams)	9	
Java Platform Module System (modules)	9	
Collection factory methods: `List.of(a, b, c); Set.of(d, e, f, g); Map.of(k1, v1, k2, v2);`	9	
Stream API improvements (`takeWhile` , `dropWhile` , `ofNullable` , `iterate` with condition)	9	
Multi-Resolution Image API	9	
Stack-Walking API	9	
`this.getClass().getPackageName()`	9	
Process API updates (detailed info about processes, e.g. ID, onExit, destroy)	9	
new methods in CompletableFuture API (`delay` , `timeout`)	9	
Interface private methods	9	
`since` and `forRemoval` in @Deprecated	9	
Interface Default and Static Methods	8	
Method References	8	
Optional class	8	
Lambda expressions	8	
Functional interfaces	8	
Stream API	8	
Effectively Final Variables	8	
Repeating Annotations	8	
New Date Time API	8	

Java language features or APIs in preview/incubation

Feature/API	**Since**	**Preview since**

Virtual threads
`Thread.startVirtualThread(runnable);`
`Thread.ofVirtual().name("duke").unstarted(runnable);`
`Executors.newVirtualThreadPerTaskExecutor();`
`Executors.newThreadPerTaskExecutor(threadFactory);`
		19

Record patterns
`record Point(int x, int y) {}`
…
`if (o instanceof Point(int x, int y)) {`
`System.out.println( x + y );`
`}`
		19

Structured concurrency
`try (var scope = new ShutdownOnFailure()) {`
`Future<String> user = scope.fork(() -> findUser());`
`Future<Integer> order = scope.fork(() -> fetchOrder());`
`}`
`//both threads are terminated here, outside of try block`
		19

Pattern Matching for switch – like `instanceof` for `switch`; `switch` is an expression and can be assigned
`String result = switch (o) {`
`case null -> null;`
`case 0 -> throw new RuntimeException("Cannot be 0"); // Special cases`
`case Integer i when i > 0 -> "Positive number";`
`case Integer i -> "Negative number"; // 0 and positive numbers handled by above rules`
`case String s -> s;`
`case Point p -> p.toString();`
`case int[] ia -> "Array length" + ia.length;`
`default -> "Something else";`
`}`
		17

Foreign Function & Memory API (an alternative to JNI)		16
Vector API – express computations that compile to optimal hardware instructions		16

New JDK tools and features since OpenJDK 8

Tool / feature	Since	Experimental since
Packaging Tool		14
Epsilon (no-op) GC		11
Experimental Java-Based JIT Compiler (Graal VM)		10
Shenandoah GC	15	12
Z GC (JEP 377)	15	11
Helpful NullPointerExceptions	14	
Launching Java files as scripts	11	
Flight recorder (data collection framework for troubleshooting)	11	
Docker Container Support: `-XX:-UseContainerSupport`	10, 8u191	
Flexible heap size selection: `-XX:MaxRAMPercentage`	10, 8u191	
Application Class-Data Sharing (CDS)	10	
jlink – custom JRE image, subset of JRE	9	
JShell (Java REPL) – run Java commands interactively	9	
Multi-Release JAR Files	9	
Compact Strings	9	

Deprecated/removed features and APIs:

Feature / API	Deprecated since	Removed since

Deprecate Finalization for Removal (`finalize()` and similar methods)	18	
Deprecate the Security Manager for Removal	17	
Constructors of primitive wrapper classes (e.g. `new Integer(1)`)	16	
ParallelScavenge + SerialOld GC Combination	14	
Applet API (Deprecated For Removal in Java 17)	9	
RMI Activation	15	17
Strongly Encapsulate JDK Internals (except sun.misc.Unsafe and some more) – internal JDK classes won't be available anymore	9	17
Nashorn JavaScript Engine	11	15
Solaris and SPARC Ports	14	15
CMS GC	9	14
Pack200 Tools and API	11	14
Java FX (moved to OpenJFX)		11
Java Web Start (was only in Oracle JDK)	9	11
Java EE and CORBA modules (JAX-WS, JAXB, JAF, Common Annotations, CORBA, JTA) Look here to find replacements for the removed modules	9	11
javah Native-Header Generator		10
jhat Heap Visualizer		9
Launch-Time JRE Version Selection		9
rt.jar and tools.jar		9
The extension mechanism (java.ext.dirs property)		9
Rarely-Used GC Combinations	8	9

For more details, the javaalmanac.io/ catalog is very useful to browse the changes in Java thoughout all its history. It can give you complete diff of APIs between selected Java versions, e.g. between Java 8 and Java 11. Very useful If you're thinking about migrating to a specific Java version.

Dávid Csákvári also wrote a similar article as mine for new features between Java 8 and Java 17, which is more detailed and with a lot of useful examples.

If you want to get started with Java 17 quickly and try out all the new features, here's a nice article to follow to get you started fast: Java 17 New Features Tutorial.