##### Java 17 and Java 8 Differences

1. **Sealed** Classes
2. **Records**
3. switch expression with support for returning values, pattern matching,
4. New HTTP Client
5. Garbage Collection Improvements

| **Feature / Aspect** | **Java 8 (2014)** | **Java 17 (2021)** |
| --- | --- | --- |
| Release Cycle | LTS version with a long support cycle. | LTS version with long-term support, more recent features. |
| Modular System (JPMS) | Not available. | Introduced in Java 9. Helps break down monolithic codebases, allowing for modularization of applications. |
| Performance | Decent performance for its time. | Significant performance improvements in garbage collection (GC), compilation, and memory handling. |
| Language Enhancements | Minimal (mainly lambdas and streams). | Pattern matching for instanceof, sealed classes, records, text blocks, switch expressions, and more. |
| Garbage Collection (GC) | Default: Parallel GC. | G1GC (default) with newer options like ZGC and Shenandoah for lower pause times and better scalability. |
| API Improvements | Some modern features like Streams and Optionals. | Enhanced APIs, new HTTP client, updated libraries, and many modern utilities. |
| Sealed Classes | Not available. | Introduced in Java 17. Allows defining a restricted hierarchy of classes. |
| Records | Not available. | Introduced in Java 16. Provides a compact syntax for immutable data classes. |
| Text Blocks | Not available. | Introduced in Java 13. Allows multiline strings, improving readability for code with large text literals. |
| Switch Expressions | Classic switch statement with fall through. | Enhanced switch expression with support for returning values, pattern matching, and simplifying code. |
| Type Inference (var) | Not available. | Introduced in Java 10. Supports local variable type inference (var). |
| Pattern Matching | Not available. | Introduced pattern matching for instanceof in Java 16, which simplifies type checks. |
| Security Features | TLS 1.2 by default. | Enhanced TLS 1.3, strong security defaults, and improved cryptography support. |
| New APIs | Basic HTTP support via HttpURLConnection. | New HttpClient API introduced in Java 11, offering modern, reactive, and asynchronous HTTP capabilities. |
| Deprecations | Older APIs still in use. | Deprecated features like RMI Activation and Applets are removed or marked for future removal. |
| Packaging Tool | No built-in packaging tool. | New jpackage tool for creating native installers (introduced in Java 14). |
| Flight Recorder | Commercial feature (available in Oracle JDK). | Now open-source and available in all JDK distributions, useful for low-overhead profiling and monitoring. |
| Preview Features | Not available. | Introduced to allow testing experimental features like Pattern Matching for Switch, Records, Text Blocks, etc. |
| Deprecation and Removal | No major removals. | Deprecated features and APIs, such as Security Manager, are being phased out. |

Key Enhancements from Java 8 to Java 17

1. Modular System (Project Jigsaw)

* Java 8: No module system, leading to monolithic applications.
* Java 17: Modules were introduced in Java 9, allowing developers to break large applications into manageable modules. This helps with encapsulation, reducing the risk of classpath conflicts.

#### Garbage Collection Improvements

* Java 8: Default garbage collector is Parallel GC, which works well for throughput but can lead to longer pause times.
* Java 17: The G1GC is the default, with options like ZGC and Shenandoah that offer low-latency garbage collection and scalable memory management.

3. Records

* Introduced in Java 16, records simplify the creation of immutable data classes by automatically generating getters, toString(), hashCode(), and equals() methods.
* Example:

java

Copy code

public record Person(String name, int age) {}

4. Text Blocks

* Available from Java 13, text blocks allow the creation of multi-line strings without needing explicit newline characters or concatenation, making it easier to handle large chunks of text.
* Example:

java

Copy code

String json = """

{

"name": "Alice",

"age": 30

}

""";

#### Switch Expressions

* In Java 8, switch was a statement, with more verbose code and the risk of fall-through. No, in **Java 8**, the switch statement **cannot return a value**.
* Java 17 enhances switch with expression syntax that can return a value, removes fall-through risk, and adds pattern matching.
* Example:

java

Copy code

String result = switch (day) {

case MONDAY, FRIDAY, SUNDAY -> "Weekend!";

case TUESDAY -> "Working day";

default -> "Unknown";

};

#### Pattern Matching for instanceof

* In Java 8, type casting was done manually after an instanceof check.
* In Java 17, pattern matching automates the casting step:

if (obj instanceof String) { String str = (String) obj; // Manual casting needed System.out.println(str.toUpperCase());

}

if (obj instanceof String str) { // Pattern matching: automatic casting System.out.println(str.toUpperCase());

}

**Pattern Matching** in Java 17 is a feature that simplifies the process of type checking and casting when working with different object types, particularly with the instanceof operator. It allows developers to streamline code by reducing boilerplate, improving readability, and eliminating the need for explicit type casting after instanceof checks.

#### Sealed Classes

* Introduced in Java 17, sealed classes restrict which classes can extend or implement them, improving security and predictability in type hierarchies.
* Example:

java

Copy code

public abstract sealed class Shape permits Circle, Rectangle {}

8. Performance and Memory Management Improvements

* Java 17 brings significant optimizations in memory management, startup time, and performance. The Z Garbage Collector (ZGC) and Shenandoah GC are optimized for low latency, making applications more scalable.

9. New HTTP Client

* Java 8: Only the outdated HttpURLConnection for making HTTP requests.
* Java 17: The modern HttpClient API (introduced in Java 11) provides a cleaner, asynchronous API for making HTTP/2 and WebSocket connections.

#### API Enhancements

* Streams and Lambdas were introduced in Java 8, but Java 17 builds on these features with more APIs, especially around the Collectors class and other utilities that improve how you work with collections, IO, and concurrency.

Conclusion:

* Java 8 was a significant release that introduced lambdas, streams, and functional programming features. It remained a staple for many developers due to its stability as an LTS version.
* Java 17, with its numerous modern features, language enhancements, improved performance, and long-term support, represents the future for enterprises and developers transitioning from Java 8. If you're still on Java 8, upgrading to Java 17 provides access to more efficient memory handling, a simplified coding style, better concurrency tools, and modern security features.