*Java 17*

***Restore Always-Strict Floating****-Point Semantics – This is mainly for scientific applications, and it makes floating-point operations consistently strict. The default floating-point operations are strict or strictfp, both of which guarantee the same results from the floating-point calculations on every platform.*

*Before Java 1.2, strictfp behavior was the default one as well. However, because of hardware issues, the architects changed, and the keyword strictfp was necessary to re-enable such behavior. So, there is no need to use this keyword anymore.*

***Enhanced Pseudo-Random Number Generators -*** *provides new interfaces and implementations for Pseudo-Random Number Generators. So, it’s easier to use different algorithms interchangeably. Legacy random classes, such as java.util.Random, SplittableRandom and SecureRandom now extend the new RandomGenerator interface.*

public IntStream getPseudoInts(String algorithm, int streamSize) { // returns an IntStream with size @streamSize of random numbers generated using the @algorithm

// where the lower bound is 0 and the upper is 100 (exclusive) return RandomGeneratorFactory.of(algorithm)

.create() .ints(streamSize, 0,100);

}

***New macOS Rendering Pipeline*** *- since Apple deprecated the OpenGL API (in macOS 10.14), used internally in Swing GUI. The new implementation uses the Apple Metal API, and apart from the internal engine, there were no changes to the existing APIs.*

***macOS/AArch64 Port (JEP 391)*** *- Apple announced a long-term plan to transition its computer line from X64 to AArch64. This JEP ports the JDK to run on AArch64 in macOS platforms.*

***Deprecate the Applet API for Removal***

***Pattern Matching for Switch (Preview) (JEP 406) -*** *This is another step toward pattern matching by enhancing pattern matching for switch expressions and statements. It reduces the boilerplate necessary to define those expressions and improves the expressiveness of the language.*

**static** **record** **Human** (String name, **int** age, String profession) {}

**public** String **checkObject**(Object obj) {

**return** **switch** (obj) {

**case** Human h -> "Name: %s, age: %s and profession: %s".formatted(h.name(), h.age(), h.profession());

**case** Circle c -> "This is a circle";

**case** Shape s -> "It is just a shape";

**case** null -> "It is null";

**default** -> "It is an object";

};

}

**public** String **checkShape**(Shape shape) {

**return** **switch** (shape) {

**case** Triangle t && (t.getNumberOfSides() != 3) -> "This is a weird triangle";

**case** Circle c && (c.getNumberOfSides() != 0) -> "This is a weird circle";

**default** -> "Just a normal shape";

};

}

***Remove the Experimental AOT and JIT Compiler*** *- The Ahead-Of-Time (AOT) compilation (JEP 295) and Just-In-Time (JIT) compiler from GraalVM (JEP-317) were features with a high cost of maintenance.*

*On the other hand, they had no significant adoption. Because of that, this JEP removed them from the platform, but developers can still leverage them using GraalVM.*

***Deprecate the Security Manager for Removal (JEP 411)*** *- The security manager aimed to secure client-side Java code is yet another feature marked for removal due to not being relevant anymore.*

***Foreign Function and Memory API (Incubator) (JEP 412)*** *- The Foreign Function and Memory API allow Java developers to access code from outside the JVM and manage memory out of the heap. With this feature, we can make a call to a C library from a Java class.*

***Vector API (Second Incubator) (JEP 414)*** *- The Vector API deals with the SIMD (Single Instruction, Multiple Data) type of operation, meaning various sets of instructions executed in parallel. It leverages specialized CPU hardware that supports vector instructions and allows the execution of such instructions as pipelines.*

*As a result, the new API will enable developers to implement more efficient code, leveraging the potential of the underlying hardware. Everyday use cases for this operation are scientific algebra linear applications, image processing, character processing, and any heavy arithmetic application or any application that needs to apply an operation for multiple independent operands.*