Java 17 features

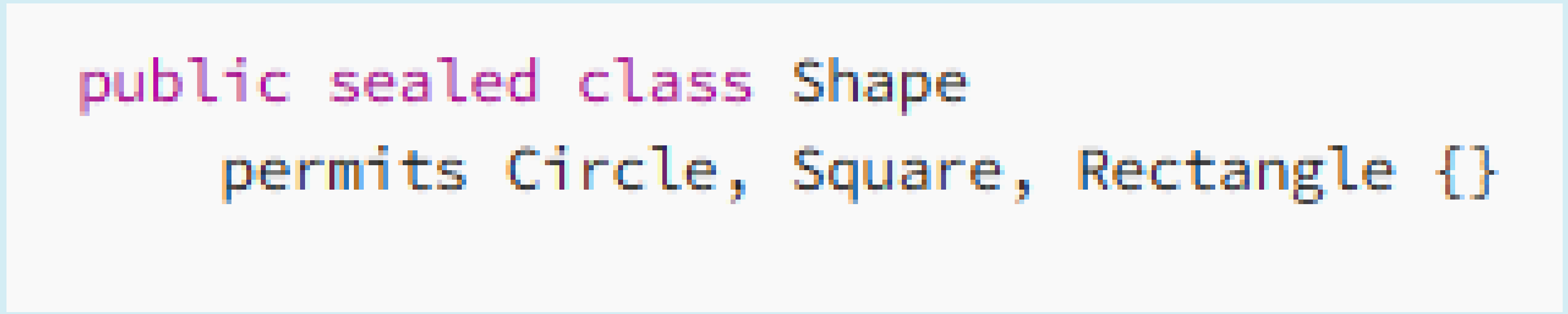
Swipe Left →

Java 17, a long-term support (LTS) release, has solidified its position as a developer’s favorite by introducing several new features and enhancements. Here’s a detailed dive into the top 10 features of Java 17, with explanations and code examples to showcase how these features can streamline development and boost productivity.

Sealed Classes

Java 17 introduces the concept of sealed classes and interfaces. This is an exciting feature for beginners as it allows you to limit the extensibility of a class or an interface. It means you can specify which other classes or interfaces are permitted to extend or implement them.

Example:



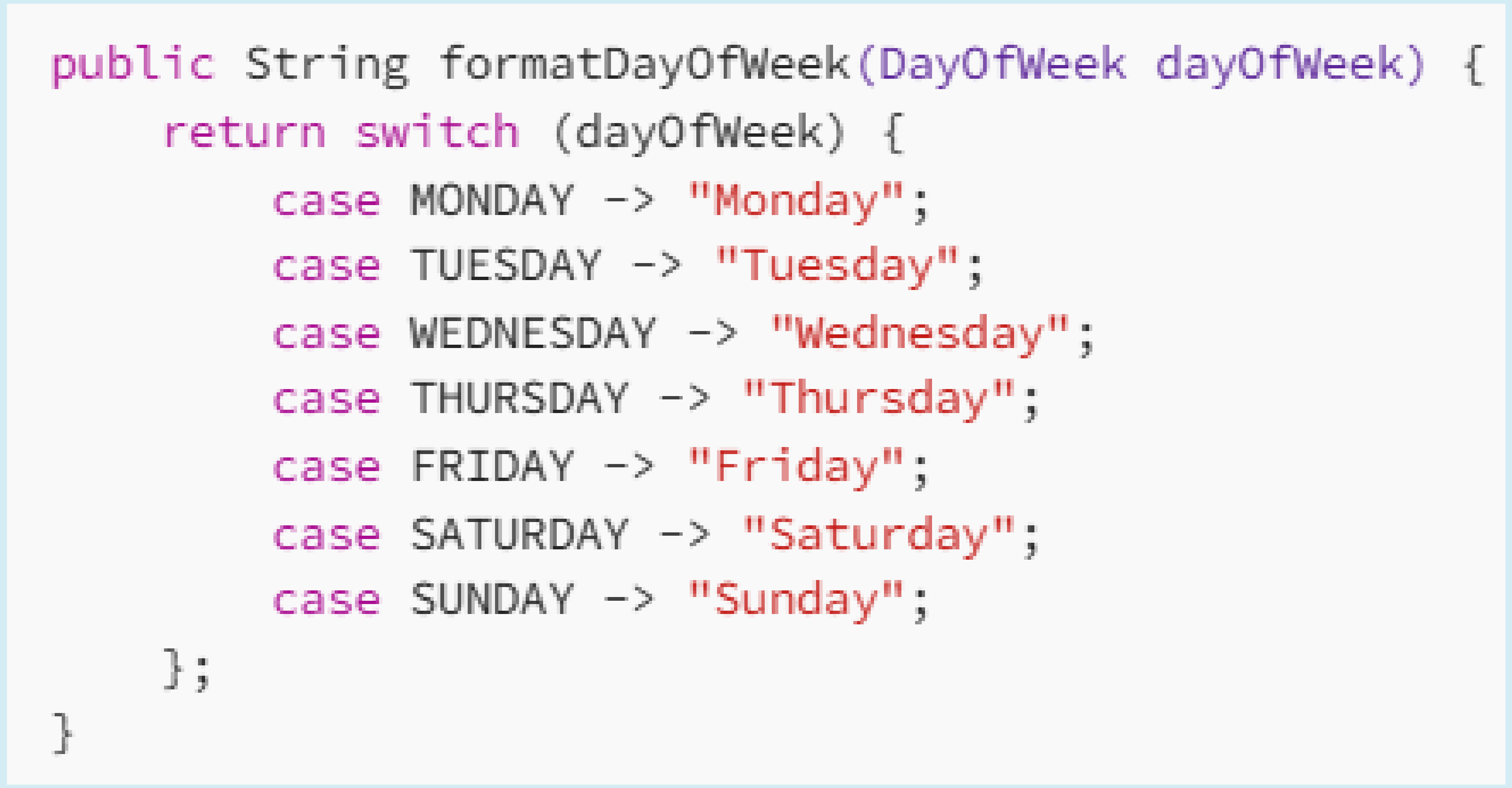
```java
public sealed class Shape
    permits Circle, Square, Rectangle {}
```

The Shape class is sealed, so it can only be extended by Circle, Square, or Rectangle. Any attempts to extend Shape with any other class will result in a compile-time error. This allows for increased control over your code and prevents misuse or unexpected behavior.

Pattern Matching for switch:

Java 17 enhances the switch statement with pattern matching capabilities. This feature simplifies the code and eliminates the need for explicit type casting, making it a great asset for beginners. Pattern matching for switch allows you to test switch expressions against a variety of patterns and execute a block of code as soon as a match is found.

Example :

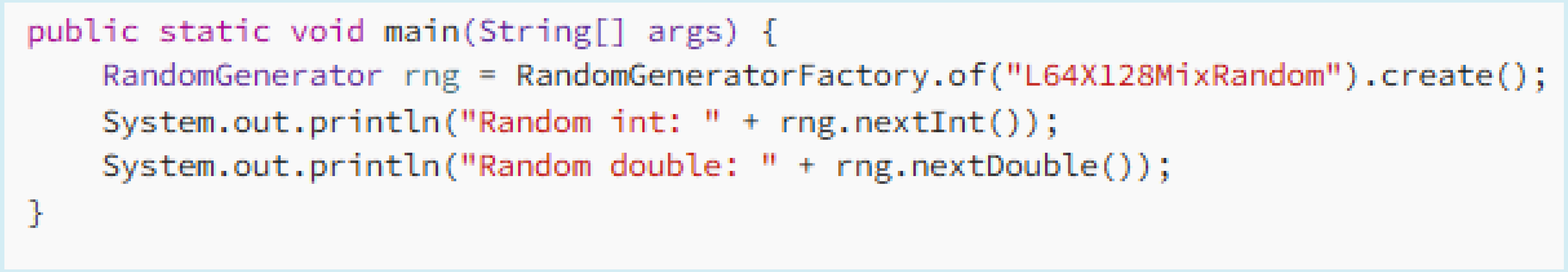


```java
public String formatDayOfWeek(DayOfWeek dayOfWeek) {
    return switch (dayOfWeek) {
        case MONDAY -> "Monday";
        case TUESDAY -> "Tuesday";
        case WEDNESDAY -> "Wednesday";
        case THURSDAY -> "Thursday";
        case FRIDAY -> "Friday";
        case SATURDAY -> "Saturday";
        case SUNDAY -> "Sunday";
    };
}
```

Enhanced Pseudo-Random Number Generators

Java 17 introduces new interfaces and implementations for random number generation, making it easier to use and extend various random number generation algorithms. The new `java.util.random` package provides the `RandomGenerator` interface, which is the base interface for all random number generators.

EXAMPLE :



```java
public static void main(String[] args) {
    RandomGenerator rng = RandomGeneratorFactory.of("L64X128MixRandom").create();
    System.out.println("Random int: " + rng.nextInt());
    System.out.println("Random double: " + rng.nextDouble());
}
```

Foreign Function & Memory API (Incubator)

Java 17 introduces an incubating Foreign Function & Memory API, which provides a pure Java API for calling native code and working with native memory. This API aims to provide a safer and more efficient alternative to the Java Native Interface (JNI).

Deprecate the Security Manager for Removal

Java 17 deprecates the Security Manager for future removal. The Security Manager has been a core component of the Java platform since its inception, but it has become less relevant over time due to the introduction of new security mechanisms like the Java Module System.

Although the Security Manager is still available in Java 17, it is marked for future removal. Developers should start migrating their applications to alternative security mechanisms.

Vector API (Second Incubator)

The Vector API moves into its second round as an incubator in Java 17. Vector operations provide a way to perform specific tasks simultaneously over large amounts of data, demonstrating significant speed-up for certain computations, an operation otherwise referred to as SIMD (Single Instruction, Multiple Data).

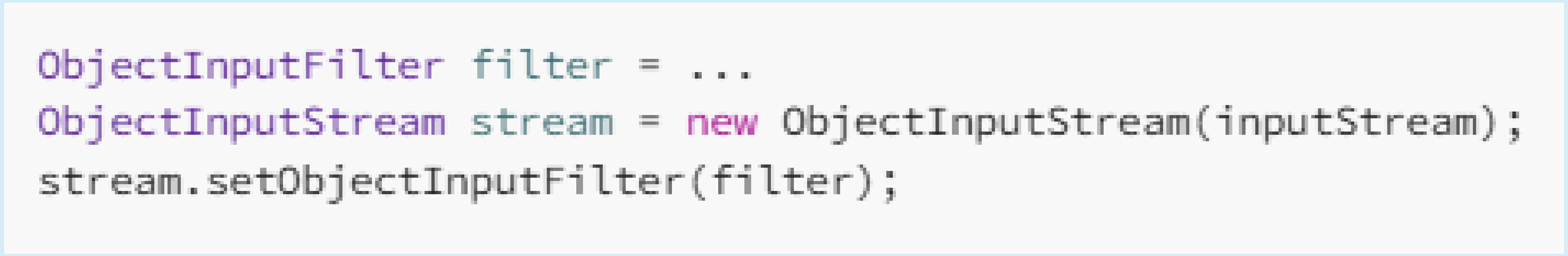
Remove the Experimental AOT and JIT Compiler

Java 11 introduced an experimental feature — the Ahead-of-Time (AOT) and Just-in-Time (JIT) compiler, based on Graal compiler. This has been removed in Java 17. Users who require AOT or JVMCI-based JIT compilation can use GraalVM instead.

Context-Specific Deserialization Filters

Deserialization of untrusted data is a known vulnerability. To mitigate this, Java 17 introduces context-specific deserialization filters. This feature allows a filter to be specified for each deserialization operation, providing contextual and finer control to filter out malicious data objects.

Example:



```
ObjectInputFilter filter = ...
ObjectInputStream stream = new ObjectInputStream(inputStream);
stream.setObjectInputFilter(filter);
```

Conclusion

Java 17 brings a variety of new features and enhancements that improve the language and platform. With additions like sealed classes, pattern matching for `switch`, new random number generators, and the macOS/AArch64 port, Java continues to evolve and adapt to modern development needs. As Java 17 is an LTS release, it’s a great time to explore these new features and consider upgrading your projects to take advantage of the latest advancements in the Java ecosystem.