Java 17 Features

1. Enhanced Pseudo-Random Generators: Creates a series of random numbers based on a deterministic algorithm.

- Introduced an improved interface which contains a unified API for all random number generators - RandomGenerator Interface. You can write code that works with any PRNG, without being tied to a specific implementation.
- Offer a **common API** that supports methods for generating different types of random data like ints(), longs(), doubles(), etc.
- It is extensible, meaning developers can implement their own custom PRNG (algorithm) by adhering to this interface - RandomGeneratorFactory
- Added new PRNGs which are algorithms to generate a sequence of random numbers
- The interface supports stream-based APIs for functional and bulk random number generation
- Deterministic Random Number Generation - the new algorithm ensures deterministic behaviour when seeded with the same initial value. This is crucial for reproducibility in scenarios like testing and simulations
- Problems with the old version was: You could only use the default PRNG algorithm (LCG in `Random`) & harder to use in functional programming pipelines

2. Pattern Matching for Switch

- Enhances pattern matching for switch expressions
- Reduces the boilerplate (repetitive code) necessary to define those expressions and improves the understanding of the language

3. Sealed Classes

- A new feature to the language
- It allows you to explicitly control which classes or interfaces can extend or implement them. This means you can create a base class or interface and restrict its subtypes to a specific, well-defined set of classes. This helps with better design, stronger type safety, and clearer intent.

4. Vector API

- Deals with SIMD operation, meaning various sets of instructions executed in parallel.
- Leverages the use of specialized CPU hardware that supports vector instructions and allows the execution of such instructions as pipelines.
- Enables developers to implement more efficient code, leveraging the potential of the underlying hardware
- Use cases: Scientific linear algebra applications, image processing, character processing and any heavy arithmetic application or any operation that needs to apply an operation for multiple independent operands.

5. Foreign Function and Memory API
   - Allows Java developers to access code from outside the JVM and manage memory out of the heap
   - Goal is to replace the JNI API and improve the security and performance compared to the old one

6. JEP 290 Context-Specific Deserialization filters
   - Enable us to validate incoming serialized data from untrusted source, a common source of many security issues. That validation happens at the JVM level, providing more security and robustness

     Before in JEP 290:
     What it does: It helps check and validate any serialized data (data that's been saved or transferred in a specific format, like files or network messages) that comes from sources you might not trust.
     Why it matters: Sometimes, attackers use malicious serialized data to hack systems. JEP 290 ensures this data is validated before being processed, making applications more secure.
     Where it happens: This validation is built directly into the Java Virtual Machine (JVM), so it works automatically at a low level to add an extra layer of protection.

     Now in JEP 415:

     What it adds: It lets developers define custom "filters" for checking serialized data. These filters can be specific to a context (like a particular part of your application) and can be chosen dynamically based on what's happening at runtime.
     How it works: Every time data is deserialized (converted back into its original form), the JVM will apply these custom filters to check if the data is safe to use.
     Why it's better: This gives applications more flexibility and fine-grained control to handle serialized data safely in different situations.

Reference:

1. https://www.baeldung.com/java-17-new-features