Java 17

Java 17, released in September 2021, is a Long-Term Support (LTS) release, offering extended support and stability.

1. Sealed Classes:

What are Sealed Classes?

Sealed classes are introduced as a standard feature in Java 17. This feature enhances the ability to model restricted hierarchies more safely and explicitly. Sealed classes allow a class or interface to control which other classes or interfaces can extend or implement it, providing better control over the inheritance structure.

Key Features of Sealed Classes

1. **Restricted Inheritance**: Allows a class or interface to explicitly specify which classes or interfaces can extend or implement it.
2. **Enhanced Safety**: Provides compile-time checks to ensure that only the permitted subclasses are used.
3. **Controlled Hierarchy**: Enables more precise control over class hierarchies, making the code easier to understand and maintain.

To define a sealed class, you use the sealed keyword and specify the permitted subclasses using the permits clause.

2. Strongly Encapsulate JDK Internals

Before Java 16: Relaxed Strong Encapsulation

Before Java 16, JDK internals were encapsulated, but there were mechanisms to relax this encapsulation, allowing access to internal APIs. Developers could access internal APIs using command-line options or reflection, potentially leading to unstable code and security risks.

Since Java 16: Strong Encapsulation by Default + Optional Relaxed Strong Encapsulation

Strong encapsulation has been the default in Java 16, but developers can still relax it using command-line options.

- **Default Strong Encapsulation**: Strong encapsulation by default improves security and stability by restricting access to internal APIs.
- **Optional Relaxation**: Developers can use command-line options to relax encapsulation if necessary, providing flexibility.

Since Java 17: Exclusively Strong Encapsulation

Since Java 17, JDK internals have been exclusively strongly encapsulated, and access to internal APIs has not been allowed. This change improves security and stability by completely restricting access to internal APIs, ensuring that developers use supported public APIs.

3. Add java.time.InstantSource

Java 17 introduces the java.time.InstantSource interface, which provides a way to access the current instant in time. This interface enhances the flexibility and usability of the java.time package.

- **Current Instant**: The InstantSource interface provides a way to access the current instant in time, enhancing the flexibility and usability of the java.time package.
- **Time Management**: This feature improves the ability to manage and work with time in Java applications.

4. Hex Formatting and Parsing Utility

Java 17 introduces a utility for formatting and parsing hexadecimal values, providing a convenient way to work with hex data.

- **Hexadecimal Data**: The utility provides methods for formatting and parsing hexadecimal values, making it easier to work with hex data in Java applications.
- **Convenience**: This feature simplifies the process of converting between integers and hexadecimal strings.

5. Context-Specific Deserialization Filters:

Java 17 introduces context-specific deserialization filters, allowing developers to specify filters for deserialization based on the context. This feature improves security by preventing deserialization vulnerabilities.

- **Security**: Context-specific deserialization filters improve security by allowing developers to specify filters for deserialization based on context, preventing deserialization vulnerabilities.
- **Flexibility**: This feature provides more control over the deserialization process, enhancing the security and stability of Java applications.

6. Enhanced Pseudo-Random Number Generators

Java 17 introduces enhanced pseudo-random number generators (PRNGs), providing more algorithms and better performance for random number generation.

- **Algorithms**: Enhanced PRNGs provide more algorithms for random number generation, improving performance and flexibility.
- **Performance**: This feature enhances the performance of random number generation, making it more efficient for various applications.

7. Performance:

Java 17 enhances the unified logging framework with support for asynchronous log flushing, improving performance by reducing the impact of logging on application execution.

- **Performance**: Asynchronous log flushing improves performance by reducing the impact of logging on application execution, making logging more efficient.
- **Efficiency**: This feature enhances the efficiency of the unified logging framework, providing better performance for Java applications.

8. Preview and Incubator Features:

Pattern matching for switch, introduced as a preview feature in Java 17, allows switch statements to match patterns, providing more concise and readable code.

- **Concise Code**: Pattern matching for switch statements allows for more concise and readable code, reducing boilerplate and improving maintainability.
- **Flexibility**: This feature provides more flexibility in handling different types and patterns in switch statements.

Foreign Function & Memory API (Incubator):

The Foreign Function & Memory API, introduced as an incubator feature in Java 17, provides a way to interact with native code and memory, enhancing interoperability and performance.

- **Interoperability**: The Foreign Function & Memory API improves interoperability with native code and memory, allowing Java applications to interact with native libraries more efficiently.
- **Performance**: This feature enhances performance by providing more direct access to native memory and functions.

Vector API (Second Incubator):

The Vector API, reintroduced as an incubator feature in Java 17, provides a way to perform vector computations in Java, improving performance for mathematical and scientific applications.

- **Vector Computations**: The Vector API provides classes and methods for performing vector computations, enhancing performance and efficiency.
- **Performance**: This feature improves the performance of applications that require intensive mathematical and scientific computations.

9. Deprecations and Deletions:

Deprecate the Applet API for Removal:

Java 17 deprecates the Applet API for removal, as applets are no longer widely used or supported. Developers are encouraged to use modern technologies, such as JavaFX or web applications, instead of applets.

Deprecate the Security Manager for Removal:

Java 17 deprecates the Security Manager for removal, as its usage has declined and modern security practices have evolved. Developers are encouraged to use modern security frameworks and practices instead of the Security Manager.

Remove RMI Activation:

Java 17 removes RMI Activation, a feature that allows remote objects to be activated on demand, as it is rarely used and has limited practical applications. Developers are encouraged to use modern alternatives for remote communication, such as RESTful APIs or gRPC.

Remove the Experimental AOT and JIT Compiler:

Java 17 removes the experimental Ahead-Of-Time (AOT) and Just-In-Time (JIT) compilers, as they are not widely used and have limited impact on performance. The experimental AOT and JIT compilers are removed, reflecting their limited use and impact on performance. Developers are encouraged to use the standard JIT compiler for performance improvements, as it is more widely used and supported.

10. Other Changes in Java 17:

New API for Accessing Large Icons:

- Java 17 introduces a new API for accessing large icons, providing a convenient way to work with high-resolution icons in Java applications.
- The new API allows developers to access and work with high-resolution icons, improving the visual quality of Java applications.
- This feature simplifies the process of working with large icons, enhancing the usability and appearance of applications.

Add support for UserDefinedFileAttributeView on macOS:

Java 17 adds support for UserDefinedFileAttributeView on macOS, allowing developers to access and manipulate user-defined file attributes on macOS systems.

System Property for Native Character Encoding Name:

- Java 17 introduces a system property for retrieving the native character encoding name, providing a convenient way to access the default character encoding of the operating system.
- This feature simplifies the process of working with character encodings, enhancing the flexibility and usability of Java applications.

Restore Always-Strict Floating-Point Semantics:

Java 17 restores always-strict floating-point semantics, ensuring that floating-point operations are consistently performed with strict precision and accuracy. This feature enhances the consistency and predictability of floating-point operations in Java applications.

New macOS Rendering Pipeline:

Java 17 introduces a new rendering pipeline for macOS, improving the performance and visual quality of Java applications on macOS systems. This feature enhances the compatibility and usability of Java applications on macOS, providing a better user experience.

macOS/AArch64 Port:

What is the macOS/AArch64 Port?

Java 17 introduces a port for macOS/AArch64, providing support for Java applications running on macOS systems with ARM-based processors.

New Page for "New API" and Improved "Deprecated" Page:

Java 17 introduces a new page for "New API" and improves the "Deprecated" page, providing better documentation and accessibility for developers.

Conclusion:

Java 17 introduces several key features and improvements, including sealed classes, enhanced encapsulation, and new APIs. These features enhance developer productivity, application performance, and the flexibility of the Java platform. By understanding and leveraging these features, developers can build more efficient and maintainable Java applications.

For a complete list of all changes in Java 17, refer to the official **Java 17 release notes**.