Java 21

Java 21 is a significant release that brings numerous enhancements, making Java even more powerful and developer-friendly. Java 21, released in September 2023, introduces several significant new features, including virtual threads, sequenced collections, and record patterns. It also includes new methods in core classes, such as String, StringBuilder, StringBuffer, Character, and Math. Additionally, this release continues to expand on preview and incubator features and brings various performance improvements and deprecations.

1. Virtual Threads – JEP 444:

What are Virtual Threads?

Virtual threads are now a permanent feature in Java 21. They are lightweight threads managed by the JVM, allowing you to create and manage thousands of threads with minimal overhead. This is part of Project Loom, which aims to simplify concurrent programming in Java.

Traditional threads can be resource-intensive and difficult to manage, especially in applications requiring thousands of concurrent operations. Virtual Threads significantly reduce the overhead associated with thread management, allowing developers to focus on business logic rather than infrastructure.

- **Lightweight**: Virtual threads are more lightweight than traditional threads, allowing for high concurrency with lower resource consumption.
- **Scalable**: They improve scalability and simplify the development of concurrent applications.

Virtual threads make it easier to write high-throughput concurrent applications without worrying about thread management.

When using virtual threads, which are lightweight, their execution can be so fast that the main thread might terminate before the virtual thread finishes its task.

2. Sequenced Collections – JEP 431:

What are Sequenced Collections?

Sequenced collections, introduced in JEP 431, maintain the order of elements as they are inserted or accessed. This feature adds interfaces to the Java Collections Framework that describe collections with a defined encounter order. By providing a standardized way to handle ordered collections, the goal is to simplify code and improve readability.

Interfaces:

- **SequencedCollection**: Extends Collection to ensure a defined order of elements.
- **SequencedSet**: Extends Set to ensure a defined order of elements without duplicates.
- **SequencedMap**: Extends Map to ensure a defined order of key-value pairs.

New Methods:

1. **SequencedCollection.reversed():** Returns a reversed view of the collection.

2. **SequencedSet.reversed():** Returns a reversed view of the set.
3. **SequencedMap.reversed()**: Returns a reversed view of the map.

Explanation:

- **Order Maintenance**: Sequenced collections maintain the order of elements, making it easier to work with ordered data structures.
- **Improved APIs**: These collections provide improved APIs for managing ordered data, simplifying code and improving readability.

3. Record Patterns – JEP 440:

Record patterns in Java 21 allow you to deconstruct record values directly within pattern-matching constructs. This feature makes it easier to extract components from records, leading to more straightforward and maintainable code.

Records, introduced in Java 16, have become a popular way to create immutable data carriers. The ability to deconstruct these records directly within pattern-matching constructs further simplifies data manipulation, especially in complex data structures.

Explanation:

- **Destructuring**: Record patterns allow developers to destructure record values, extracting their components in a more readable way.
- **Concise Code**: This feature reduces boilerplate code, making it easier to work with record values.

4. Pattern Matching for switch – JEP 441:

Pattern matching for switch, introduced in JEP 441, allows switch statements to match patterns, providing more concise and readable code.

Explanation:

- **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.

5. New Methods in String, StringBuilder, StringBuffer, Character, and Math:

Java 21 introduces several new methods across the core classes String, StringBuilder, StringBuffer, Character, and Math. These methods, though not always highlighted in the JEPs or release notes, can be found in the API documentation and provide useful enhancements for developers. Let’s explore these new methods with examples and their outputs.

1. New Methods in String:

a) String.indexOf(String str, int beginIndex, int endIndex): This method searches for the specified substring within a subrange of the string, starting from beginIndex and ending at endIndex.

b) String.indexOf(char ch, int beginIndex, int endIndex): This method searches for the specified character within a subrange of the string, starting from beginIndex and ending at endIndex.

c) String.splitWithDelimiters(String regex, int limit): This method splits the string at substrings matched by the regular expression and returns an array of all parts, including the splitting strings. The string is split at most limit-1 times.

Both StringBuilder and StringBuffer have been enhanced with the following methods:

a) repeat(CharSequence cs, int count): This method appends the specified CharSequence to the StringBuilder or StringBuffer a specified number of times.

b) repeat(int codePoint, int count): This method appends the specified Unicode code point or character to the StringBuilder or StringBuffer a specified number of times.

The Character class in Java 21 includes several new methods that focus on emoji support:

a) isEmoji(int codePoint): This method checks if the specified Unicode code point is an emoji.

b) isEmojiComponent(int codePoint): This method checks if the specified Unicode code point is a component used in constructing emojis.

c) isEmojiModifier(int codePoint): This method checks if the specified Unicode code point is an emoji modifier, used to modify the appearance of emojis.

d) isEmojiModifierBase(int codePoint): This method checks if the specified Unicode code point can be a base for emoji modifiers.

e) isEmojiPresentation(int codePoint): This method checks if the specified Unicode code point is an emoji presentation, meaning it is typically displayed as an emoji.

f) isExtendedPictographic(int codePoint): This method checks if the specified Unicode code point is part of the extended pictographic symbols, which include many emojis.

Java 21 adds a new clamp() method to the Math class, which ensures that a number is within a specified range.

a) Math.clamp(int value, int min, int max): This method checks if the value is within the range [min, max]. If the value is less than min, it returns min; if it is greater than max, it returns max.

b) Math.clamp(long value, long min, long max): This method works similarly but for long values.

c) Math.clamp(double value, double min, double max): This method clamps a double value within a specified range.

d) Math.clamp(float value, float min, float max): This method works similarly, but for float values.

6. Preview and Incubator Features:

String Templates (Preview) – JEP 430:

String templates are a new feature in Java 21 that allows you to embed expressions directly within string literals. This feature simplifies string manipulation by eliminating the need for complex concatenation or formatting operations.

String manipulation is a common programming task, but it often involves verbose and error-prone code. String templates make it easier to create and manage strings, especially when working with dynamic content.

- Unnamed patterns and variables allow for the use of patterns and variables without specifying a name.
- Unnamed classes and instance main methods simplify the creation of classes and instance main methods without specifying names.
- Scoped values provide a way to define values that are scoped to a specific thread or task, improving the management of context-specific data.
- Structured concurrency simplifies the handling of concurrent tasks by grouping related tasks and managing their lifecycle as a unit.

The Foreign Function & Memory API 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 (Sixth Incubator) – JEP 448:

The Vector API 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.

7. Other Changes in Java 21:

Generational ZGC – JEP 439:

Generational ZGC improves the performance of the Z Garbage Collector by adding generational support.

- Generational Support: Generational ZGC enhances the performance of the Z Garbage Collector by adding generational support, improving the efficiency of garbage collection.
- Performance: This feature improves the performance of applications by reducing the overhead associated with garbage collection.

Generational Shenandoah (Experimental) – JEP 404:

Generational Shenandoah improves the performance of the Shenandoah Garbage Collector by adding generational support.

- Generational Support: Generational Shenandoah enhances the performance of the Shenandoah Garbage Collector by adding generational support, improving the efficiency of garbage collection.
- Performance: This feature improves the performance of applications by reducing the overhead associated with garbage collection.

Deprecate the Windows 32-bit x86 Port for Removal – JEP 449:

Java 21 deprecates the Windows 32-bit x86 port for removal, reflecting the declining use of 32-bit systems and encouraging developers to use 64-bit systems.

- Deprecated: The Windows 32-bit x86 port is deprecated for removal, encouraging developers to use 64-bit systems for better performance and compatibility.
- Modern Systems: Developers are encouraged to use modern 64-bit systems, reflecting the declining use of 32-bit systems.

Prepare to Disallow the Dynamic Loading of Agents – JEP 451:

Java 21 prepares to disallow the dynamic loading of agents, improving the security and stability of the JVM.

- Security: Disallowing the dynamic loading of agents improves the security and stability of the JVM, reducing the risk of unauthorized modifications.
- Stability: This feature enhances the stability of Java applications by preventing the dynamic loading of agents.

Key Encapsulation Mechanism API – JEP 452

The Key Encapsulation Mechanism API provides a way to encapsulate cryptographic keys, improving the security and flexibility of key management.

- Key Encapsulation: The Key Encapsulation Mechanism API provides a way to encapsulate cryptographic keys, improving the security and flexibility of key management.
- Security: This feature enhances the security of Java applications by providing a standardized way to manage cryptographic keys.

Improve Thread.sleep(millis, nanos) For Sub-millisecond Granularity:

Java 21 improves the Thread.sleep(millis, nanos) method to support sub-millisecond granularity, providing more precise control over thread sleeping.

- Sub-millisecond Granularity: The improvement to Thread.sleep(millis, nanos) provides more precise control over thread sleeping, allowing for better performance and timing accuracy.
- Performance: This feature enhances the performance of applications that require precise timing and control over thread sleeping.

Last-Ditch Full GC Should Also Move Humongous Objects:

Java 21 changes the behavior of the last-ditch full garbage collection (GC) to also move humongous objects, improving memory management and performance.

This feature enhances memory management by ensuring that all objects, including humongous ones, are properly managed during garbage collection.

Conclusion:

Java 21 introduces several key features and improvements, including virtual threads, sequenced collections, and record patterns. 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 21, refer to the official Java 21 release notes.