Java 21 Interview Questions

What are the new features introduced in Java 21?

- Record Patterns: Allows deconstructing records using patterns.
- Pattern Matching for switch: Enhances switch expressions with pattern matching.
- Virtual Threads: Introduces lightweight threads to improve concurrency.
- String Templates: Provides a new way to handle string formatting.
- Sealed Interfaces: Expands the use of sealed types to interfaces.
- Value Types (preview): Introduces a new way to represent immutable types for better performance and memory efficiency.
- Foreign Function & Memory API (API Incubator): Improves interaction with native code and memory.

Explain the enhancements to the Record class.

- Record Patterns: You can now use record patterns to destructure records directly within pattern matching constructs. This simplifies working with records and improves code readability.
- Enhanced Equality and Hash Code Methods: Improvements have been made to ensure better consistency and efficiency in methods like equals and hashCode.

How has Pattern Matching been improved in Java 21?

Java 21 introduces Pattern Matching for switch, which allows for more expressive and flexible switch expressions and statements. It simplifies complex switch statements by allowing patterns to be used in case labels, enabling a more concise and readable way to handle different types of input.

What are the new capabilities of Virtual Threads in Java 21?

Virtual Threads are lightweight threads that aim to simplify concurrency by allowing you to create and manage thousands of concurrent tasks with minimal overhead. They use a virtualized stack and are managed by the Java runtime, making them more efficient compared to traditional platform threads. Virtual threads improve scalability and reduce the complexity associated with traditional thread management.

Discuss the updates to Sealed Classes in Java 21.

Java 21 extends Sealed Classes to interfaces, allowing you to control which classes or interfaces can implement or extend a sealed interface. This improves the safety and maintainability of type hierarchies by restricting subclassing and ensuring that all possible subclasses are known and accounted for.

How does Project Loom impact concurrency in Java 21?

Project Loom introduces Virtual Threads, which significantly impact concurrency by making it easier to write and manage concurrent code. Virtual Threads enable high-throughput applications with simpler and more manageable concurrency models, reducing the overhead and complexity of traditional thread-based concurrency.

What improvements have been made to the Foreign Function & Memory API in Java 21?

- Improved APIs for interacting with native code: New methods and features make it easier to allocate and manage native memory, interact with native libraries, and handle native data structures.
- Better support for performance and safety: Enhancements have been made to improve performance and safety when dealing with foreign functions and memory.

Explain the new String Templates feature in Java 21.

String Templates provide a new way to create and manage strings with embedded expressions. Instead of using traditional string concatenation or formatting, you can use a template syntax to embed variables and expressions directly within the string, making code more readable and reducing the risk of errors.

How does Value Types in Java 21 enhance performance?

Value Types are a new feature aimed at improving performance and memory efficiency by providing a way to define immutable types that can be optimized by the JVM. Unlike traditional reference types, value types can be stored directly in arrays and objects, reducing the overhead of object allocation and improving cache locality.

What are the benefits of Pattern Matching for Switch introduced in Java 21?

Pattern Matching for Switch allows for more expressive and concise switch statements by enabling patterns to be used in case labels. This feature simplifies complex switch logic, improves code readability, and reduces the need for nested switch or if-else statements.

Detailed Feature Explanations

How does the Record class in Java 21 differ from earlier versions?

Records in Java 21 now support Record Patterns, allowing them to be used in pattern matching expressions. This enhancement simplifies the process of destructuring records, making it easier to access and manipulate individual components.

Describe the new syntax and capabilities introduced in Pattern Matching for instanceof in Java 21.

Pattern Matching for instanceof introduces a new syntax that allows you to test and cast objects in a single operation. Instead of:

```java
if (obj instanceof String) {
  String s = (String) obj;
}
```

You can now use:

```java
if (obj instanceof String s) {
  // Use s directly
}
```

This feature simplifies type checks and casts, reducing boilerplate code and improving readability.

What are Virtual Threads, and how do they improve scalability in Java 21?

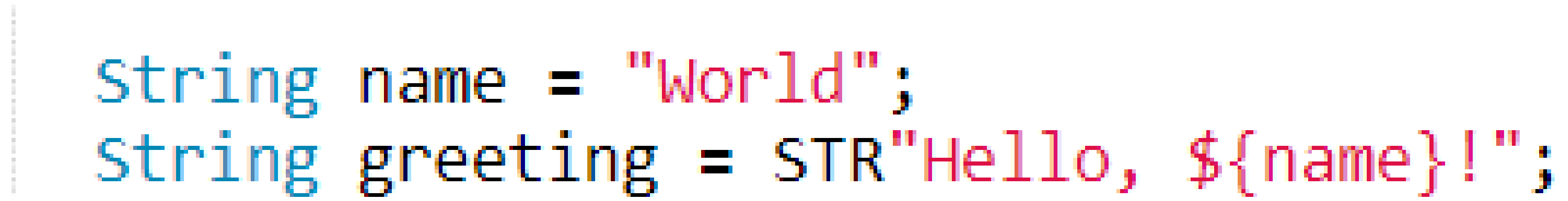
Virtual Threads are lightweight threads that reduce the overhead associated with traditional threads. They are managed by the JVM rather than the operating system, allowing you to create a large number of concurrent tasks with minimal resource consumption. This improvement enhances scalability by making it easier to handle high levels of concurrency without significant performance degradation.

How does Java 21 improve the usability of Sealed Interfaces and Sealed Classes?

Java 21 extends the concept of Sealed Classes to interfaces, allowing you to restrict which classes or interfaces can implement or extend a sealed interface. This feature provides greater control over type hierarchies, improves safety, and ensures that all possible subclasses are known, making code more maintainable and easier to reason about.

How does String Templates simplify string handling and formatting in Java 21?

String Templates allow you to embed expressions directly within string literals using a new template syntax. This feature simplifies string formatting and reduces the need for concatenation or complex formatting methods. For example:



```java
String name = "World";
String greeting = STR"Hello, ${name}!";
```

This makes it easier to create and manage strings with embedded variables and expressions.

Explain how Value Types can be used to optimize performance and memory usage in Java 21.

Value Types are designed to be immutable and can be stored directly in memory rather than as objects. This reduces the overhead of object allocation and improves cache locality. Value types are especially useful for representing simple, immutable data structures that do not require object identity.

What is the significance of Pattern Matching for Switch in Java 21, and how does it affect code readability and maintenance?

Pattern Matching for Switch allows patterns to be used in switch statements, making it easier to handle complex conditional logic. It improves code readability by reducing boilerplate code and eliminating the need for nested switch or if-else statements. This feature simplifies maintaining and updating switch statements.

Advanced Java 21 Concepts

Discuss the impact of Project Loom on traditional Java concurrency models.

Project Loom impacts traditional concurrency models by introducing Virtual Threads, which simplify concurrent programming and improve scalability. Unlike traditional platform threads, which are limited by system resources, virtual threads allow for the creation of a large number of concurrent tasks with minimal overhead. This makes it easier to write high-performance, concurrent applications.

How can you leverage Virtual Threads to improve the performance of I/O-bound applications in Java 21?

Virtual Threads are particularly useful for I/O-bound applications because they allow you to handle many concurrent I/O operations with minimal overhead. Since virtual threads are lightweight, you can create thousands of them to handle concurrent I/O operations efficiently. This reduces the need for complex thread management and improves overall performance.

What are the potential use cases and benefits of Value Types in modern Java applications?

- Representing immutable data structures: Value types are ideal for representing simple, immutable data that does not require identity.
- Improving performance: Value types can reduce the overhead of object allocation and improve cache efficiency.
- Optimizing memory usage: By avoiding the overhead of reference types, value types can help reduce memory consumption.

Thank You!!