Java - JDK 21

Released on:
September 19, 2023

By Theepana Govintharajah

Features

- **Virtual threads**
- **Record patterns**
- **Sequenced collections**
- **Key encapsulation mechanism API (KEM)**
- **Generational ZGC**
- **String templates** (Preview)
- **Structured concurrency** (Preview)
- **Unnamed Classes and Instance Main Methods** (Preview)
- **Scoped Values** (Preview)

By Theepana Govintharajah

Virtual threads

- Virtual threads are a new type of thread that is managed by the JVM.
- They are lightweight and can be used to improve the performance of concurrent applications.
- They were introduced as a preview feature in JDK 19 and finalized in JDK 21.

By Theepana Govintharajah

Record patterns

- Record patterns are a new type of pattern that can be used to match and decompose record values.

- They make it easier to write concise and readable code.

(Record values are a type of data structure that was introduced in Java 14.)

Sequenced collections

- Sequenced collections are a new type of collection that maintains the defined order in which elements are encountered.

- They can be used to implement efficient and reliable concurrent data structures.

- Sequenced collections are defined by the new ***SequencedCollection*** interface.

By Theepana Govintharajah

Sequenced collections

- This interface provides methods for adding, removing, and retrieving elements at both ends of the collection.

- It also provides a method for getting a reversed view of the collection.

By Theepana Govintharajah

Sequenced collections

- Sequenced collections can be used to implement a variety of concurrent data structures, such as queues, stacks, and deques.

- They can also be used to implement efficient algorithms that operate on ordered data, such as sorting and searching.

By Theepana Govintharajah

Key encapsulation mechanism API (KEM)

- It is a new API that provides a standardized way to encapsulate cryptographic keys.
- This can help to improve the security of Java applications.

By Theepana Govintharajah

- A KEM is a cryptographic algorithm that is used to generate a shared secret key between two parties.

- The sender uses the receiver's public key to generate a key encapsulation message (KEM), which is sent to the receiver. The receiver then uses their private key to decapsulate the KEM to recover the shared secret key.

Generational ZGC

- It is a new type of garbage collector that was introduced in JDK 21.

- Generational ZGC is still under development, but it has shown promising results in benchmarks. It is expected to become the default garbage collector in future releases of the Java SE platform.

By Theepana Govintharajah

Generational ZGC

- Generational ZGC performs most of its work concurrently with the application running, which minimizes the impact on application performance.

- Generational ZGC can use multiple threads to collect garbage in parallel, which can improve scalability on multi-core systems.

By Theepana Govintharajah

String templates

- String templates are a new preview feature in JDK 21 that provides a way to format strings using expressions.

- They can be used to write more concise and readable code.

- String templates are defined using the ${} syntax.

By Theepana Govintharajah

String templates

Code

```
String template = "Hello, ${name}! You are ${age} years old.";

String result = StringTemplate.format(template, "Alice", 25);

System.out.println(result);
```

Output:

Hello, Alice! You are 25 years old.

String templates can also be used to format more complex data structures, such as lists and objects.

Code:

```
${names[0]} and ${names[1]} are friends.
```

By Theepana Govintharajah

Structured concurrency

- Structured concurrency is a new preview feature in JDK 21 that provides a way to simplify concurrent programming.

- It treats groups of related tasks running in different threads as a single unit of work.

- This streamlines error handling and cancellation, improves reliability, and enhances observability.

By Theepana Govintharajah

Structured concurrency

- Structured concurrency is based on the concept of a scope.
- A scope is a unit of work that encompasses a group of related tasks.
- Scopes can be nested, which allows you to create hierarchical concurrent programs.

By Theepana Govintharajah

Unnamed Classes and Instance Main Methods

- Unnamed classes and instance main methods are a new preview feature in JDK 21 that allows you to write simpler and more concise scripts in Java.

- Unnamed classes are classes that do not have a name.

By Theepana Govintharajah

- They are created using the class keyword without a following identifier.

- Unnamed classes can be used to encapsulate code that is only needed in a single place, such as in a script.

- It can be used to make scripts more modular and reusable, as they allow you to create scripts that can be reused in different contexts.

By Theepana Govintharajah

Scoped Values

- Scoped values are a new preview feature in JDK 21 that allows you to safely and efficiently share data between methods in a large program without resorting to method arguments.

- They are similar to thread-local variables, but they are more efficient and can be used in a wider variety of situations.

By Theepana Govintharajah

- The value will be accessible to all methods in the current scope, but it will not be accessible to methods outside of the scope.

- Scoped values can also be used to avoid passing unnecessary data to methods. For example, if you have a method that needs to access the current user ID, you can use a scoped value to bind the user ID to the method. This will avoid having to pass the user ID to the method explicitly.

By Theepana Govintharajah

```
public class ScopedValuesExample {
    private static final ScopedValue<String> USER_ID  = new ScopedValue<>();

    public static void main(String[] args) {
            // Bind a user ID to the scoped value.
            USER_ID.bind("1234567890");
            // Call a method that uses the scoped value.
            useUserId();
            // Unbind the user ID from the scoped value.
            USER_ID.unbind();
    }

    private static void useUserId() {
            // Get the user ID from the scoped value.
            String userId = USER_ID.get();
            // Use the user ID.
            System.out.println("The user ID is: " + userId);
    }
}
```

By Theepana Govintharajah

Follow me for more similar posts

By Theepana Govintharajah