Overview of Java Structured Concurrency

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Lesson

• Understand the Java structured concurrency model

• This model is designed to enable the processing of "embarrassingly parallel" tasks atop the virtual threading mechanisms available in Java 19 (& beyond)
Overview of Java
Structured Concurrency
• Structured concurrency was added recently to Java as a concurrent programming paradigm

**Overview of Java Structured Concurrency**

**JEP 428: Structured Concurrency (Incubator)**

- **Authors**: Alan Bateman, Ron Pressler
- **Owner**: Alan Bateman
- **Type**: Feature
- **Scope**: JDK
- **Status**: Closed/Delivered
- **Release**: 19
- **Component**: core-libs
- **Discussion**: loom dash dev at openjdk dot java dot net
- **Reviewed by**: Alex Buckley, Brian Goetz
- **Created**: 2021/11/15 15:01
- **Updated**: 2022/08/10 15:58
- **Issue**: 8277129

**Summary**

Simplify multithreaded programming by introducing an API for *structured concurrency*. Structured concurrency treats multiple tasks running in different threads as a single unit of work, thereby streamlining error handling and cancellation, improving reliability, and enhancing observability. This is an incubating API.

**Goals**

- Improve the maintainability, reliability, and observability of multithreaded code.
- Promote a style of concurrent programming which can eliminate common risks arising from cancellation and shutdown, such as thread leaks and cancellation delays.

See [openjdk.org/jeps/428](http://openjdk.org/jeps/428)
Structured concurrency was added recently to Java as a concurrent programming paradigm.

It’s intended to make programs easier to read & understand, quicker to write, & safer.

See [en.wikipedia.org/wiki/Structured_concurrency](en.wikipedia.org/wiki/Structured_concurrency)
Overview of Java Structured Concurrency

- Structured concurrency was added recently to Java as a concurrent programming paradigm.
- It’s intended to make programs easier to read & understand, quicker to write, & safer.
- “Safer” avoids thread leaks & orphan threads.

See [en.wikipedia.org/wiki/Orphan_process](en.wikipedia.org/wiki/Orphan_process)
Structured concurrency was added recently to Java as a concurrent programming paradigm.

- It’s intended to make programs easier to read & understand, quicker to write, & safer.
- “Safer” avoids thread leaks & orphan threads.

**Overview of Java Structured Concurrency**

The lifetime of Thread $T_1$ & Thread $T_2$ are constrained to the enclosing scope.
Overview of Java Structured Concurrency

Java structured concurrency makes the start & end of concurrent code explicit.

```java
try (var scope = new StructureTaskScope.ShutdownOnFailure()) {
    var results = new ArrayList<Future<BigFraction>>()

    for (var bigFraction : generateRandomBigFractions(count))
        results.add(scope
            .fork(() ->
                reduceAndMultiply(bigFraction,
                    sBigReducedFraction));

    scope.join();

    sortAndPrintList(results);
}
```

We will walk through this example quickly now & will explore it in detail later on.

See [github.com/douglas craig schmidt/LiveLessons/tree/master/Loom/ex3](https://github.com/douglas craig schmidt/LiveLessons/tree/master/Loom/ex3)
Overview of Java Structured Concurrency

- Java structured concurrency makes the start & end of concurrent code explicit

```java
try (var scope = new StructureTaskScope.ShutdownOnFailure()) {
    var results = new ArrayList<>(new ArrayList<Future<BigFraction>>())

    for (var bigFraction : generateRandomBigFractions(count))
        results.add(scope.fork(() ->
            reduceAndMultiply(bigFraction, sBigReducedFraction));

    scope.join();

    sortAndPrintList(results);
}
```

Define a scope for splitting a task into concurrent subtasks
try (var scope = new StructureTaskScope.ShutdownOnFailure()) {
    var results = new ArrayList<Future<BigFraction>>()
    for (var bigFraction : generateRandomBigFractions(count))
        results.add(scope.fork() -> reduceAndMultiply(bigFraction, sBigReducedFraction));
    scope.join();
    sortAndPrintList(results);}

Overview of Java Structured Concurrency

- Java structured concurrency makes the start & end of concurrent code explicit

Start new virtual threads to reduce/multiply BigFraction objects concurrently
try (var scope = new StructureTaskScope.ShutdownOnFailure()) {
    var results = new ArrayList<F<BigFraction>>()
    for (var bigFraction : generateRandomBigFractions(count))
        results.add(scope.fork(() ->
            reduceAndMultiply(bigFraction,
                sBigReducedFraction));
    scope.join();
    sortAndPrintList(results);
}
Java structured concurrency makes the start & end of concurrent code explicit:

```java
try (var scope = new StructureTaskScope.ShutdownOnFailure()) {
    var results = new ArrayList<Future<BigFraction>>()
    for (var bigFraction: generateRandomBigFractions(count))
        results.add(scope.fork(() -> reduceAndMultiply(bigFraction, sBigReducedFraction));

    scope.join();

    sortAndPrintList(results);
}
```

The close() method of `scope` is called automatically when this block of code exits.
Overview of Java Structured Concurrency

- Java structured concurrency provides several guarantees
Overview of Java Structured Concurrency

- Java structured concurrency provides several guarantees
- When a program’s flow of control is split into multiple threads these threads always complete at the end of a flow

See theboreddev.com/understanding-structured-concurrency
Overview of Java Structured Concurrency

- Java structured concurrency provides several guarantees
- When a program’s flow of control is split into multiple threads these threads always complete at the end of a flow

All these threads must complete by the end of the enclosing scope
Overview of Java Structured Concurrency

- Java structured concurrency provides several guarantees
  - When a program’s flow of control is split into multiple threads these threads always complete at the end of a flow
  - No “orphaned threads” occur in an application
Overview of Java Structured Concurrency

- Java structured concurrency provides several guarantees
  - When a program’s flow of control is split into multiple threads these threads always complete at the end of a flow
  - No “orphaned threads” occur in an application
  - This paradigm is designed to mimic structured programming

See auroratide.com/posts/understanding-kotlin-coroutines
Overview of Java Structured Concurrency

- Java structured concurrency is intended for “embarrassingly parallel” programs

“Embarrassingly parallel” tasks have little/no dependency or need for communication between tasks or for sharing results between them.

See [en.wikipedia.org/wiki/Embarrassingly_parallel](en.wikipedia.org/wiki/Embarrassingly_parallel)
Overview of Java Structured Concurrency

- Java structured concurrency is intended for “embarrassingly parallel” programs
- e.g., interacting with many micro-services in a cloud computing environment

See en.wikipedia.org/wiki/Microservices
End of Overview of Java
Structured Concurrency