Applying Java Structured Concurrency: Case Study ex2

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 Java’s structured concurrency model
• Recognize the classes used to program Java’s structure concurrency model, e.g.
  • ThreadPerTaskExecutor
    • Case study ex2 shows how Java Executors is updated with new factory methods that create a (virtual) Thread per task

```java
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
    return integers
        .stream()
        .map(primeCandidate ->
              checkPrimality(primeCandidate, executor))
        .toList();
}
```

The tasks in this case study are all CPU-bound
Learning Objectives in this Part of the Lesson

• Understand Java’s structured concurrency model
• Recognize the classes used to program Java’s structure concurrency model, e.g.
  • ThreadPerTaskExecutor
    • Case study ex2 shows how Java Executors is updated with new factory methods that create a (virtual) Thread per task
  • It also shows how to combine Java Streams with the new Java Executors features

```java
try (ExecutorService executor = Executors.
    newVirtualThreadPerTaskExecutor())
{
    return integers.
        stream()

        .map(primeCandidate ->
            checkPrimality(primeCandidate, executor))

        .toList();
}
```
Applying Java Structured Concurrency to Case Study ex2
private static List<Future<PrimeResult>> checkPrimalities
    (List<Integer> integers) {
        // Create a new scope to execute virtual tasks, which exits
        // only after all tasks complete by using the new AutoClosable
        // feature of ExecutorService in conjunction with a
        // try-with-resources block.
        try (ExecutorService executor =
            Executors.newVirtualThreadPerTaskExecutor()) {
            return integers
            .stream()
            .map((mapper: primeCandidate ->
                // Use executor to start a virtual thread.
                checkPrimality(primeCandidate, executor))
            .toList();
        }
    }
End of Applying Java Structured Concurrency: Case Study ex2