Applying Java Structured Concurrency: Case Study ex3

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
  - StructuredTaskScope
    - Case study ex3 shows how to program the Java Structured TaskScope subclasses
      - Both ShutdownOnFailure & ShutdownOnSuccess

```
try (var scope = new
       StructuredTaskScope
       .ShutdownOnFailure()) {
  var results = ...
  for (var bf :
         generateRandomBigFractions
           (count))
    results.add
      (scopes.fork(...));

  scope.join();

  sortAndPrintList(results);
}
```

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
  - StructuredTaskScope
    - Case study ex3 shows how to program the Java Structured TaskScope subclasses
    - It uses "Classic Java" features rather than Java streams

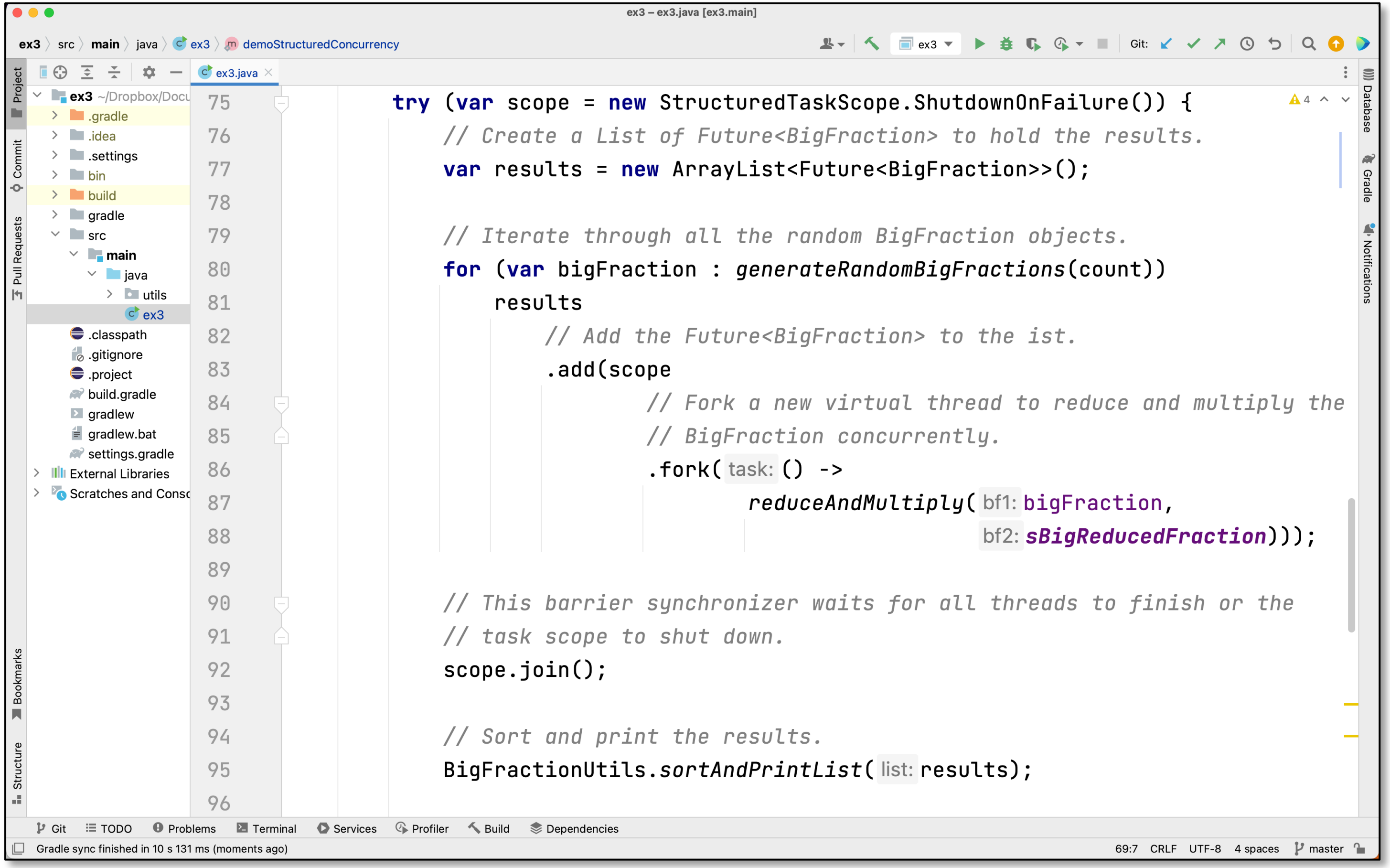
```
try (var scope = new
          StructuredTaskScope
          .ShutdownOnFailure()) {
    var results = ...
    for (var bf :
           generateRandomBigFractions
             (count))
      results.add
         (scopes.fork(...));

    scope.join();

    sortAndPrintList(results);
}
```

Applying Java Structured Concurrency to Case Study ex3

Applying Java Structured Concurrency to Case Study ex3



```
        try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
            // Create a List of Future<BigFraction> to hold the results.
            var results = new ArrayList<Future<BigFraction>>();

            // Iterate through all the random BigFraction objects.
            for (var bigFraction : generateRandomBigFractions(count))
                results
                    // Add the Future<BigFraction> to the ist.
                    .add(scope
                        // Fork a new virtual thread to reduce and multiply the
                        // BigFraction concurrently.
                        .fork( task: () ->
                            reduceAndMultiply( bf1: bigFraction,
                                               bf2: sBigReducedFraction)));

            // This barrier synchronizer waits for all threads to finish or the
            // task scope to shut down.
            scope.join();

            // Sort and print the results.
            BigFractionUtils.sortAndPrintList( list: results);

```

Gradle sync finished in 10 s 131 ms (moments ago)

See github.com/douglascraigschmidt/LiveLessons/tree/master/Loom/ex3

End of Applying Java Structured Concurrency: Case Study ex3