Applying Java Structured Concurrency: Case Study ex4 (Part 3)

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

- Case study ex4 compares & contrasts the programming models & performance results of Java parallel streams, completable futures, Project Reactor, RxJava, & Java structured concurrency frameworks when applied to download, transform, & store many images from a remote web server

```
Flux
   .fromIterable(getUrlList())
   .parallel()
   .runOn(Schedulers
            .boundedElastic())
   .map(...::downloadImage)
   .flatMap(...::transformImage)
   .map(...::storeImage)
   .sequential()
   .collectList()
   .block();
```

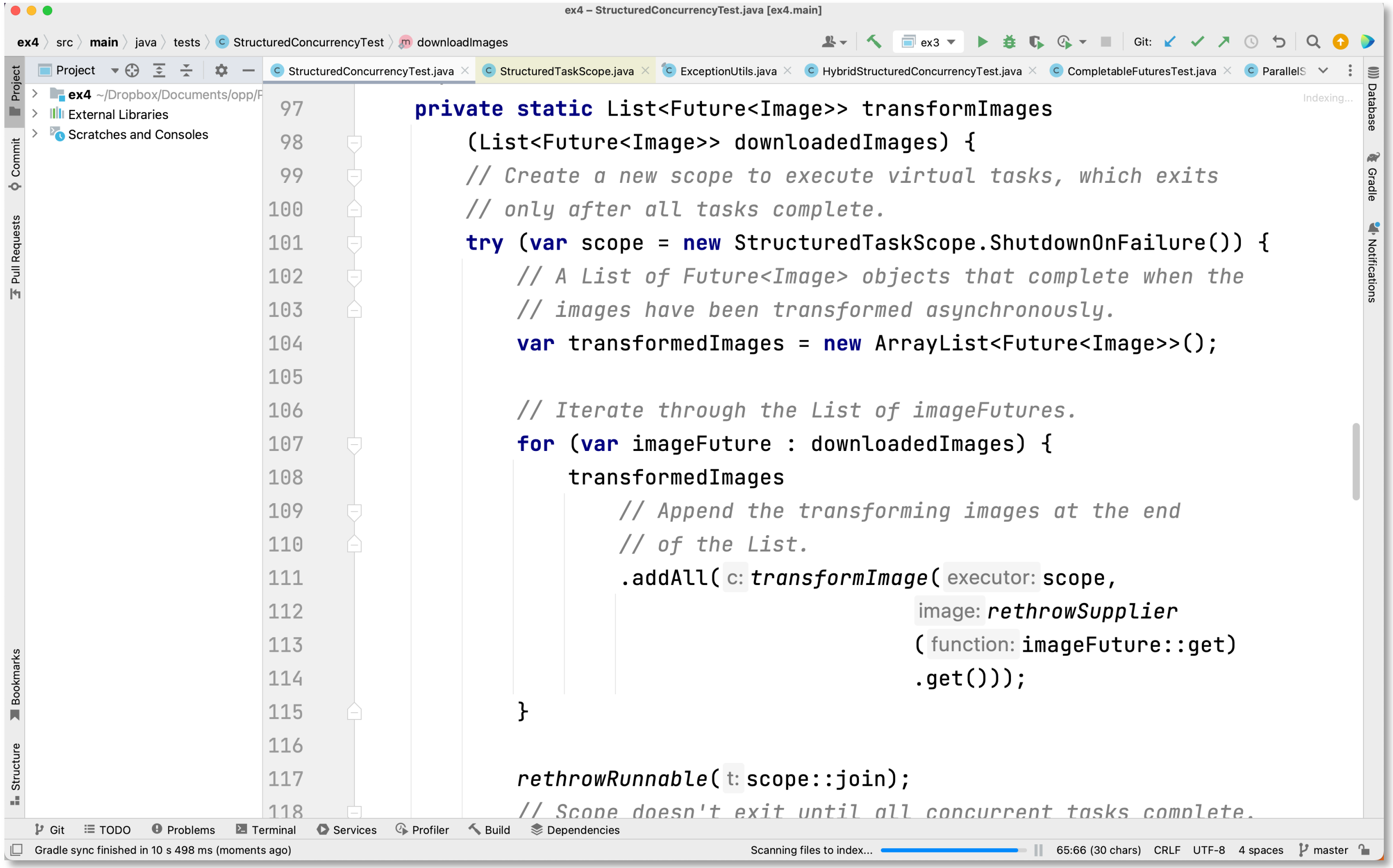
Learning Objectives in this Part of the Lesson

- Case study ex4 compares & contrasts the programming models & performance results of Java parallel streams, completable futures, Project Reactor, RxJava, & Java structured concurrency frameworks when applied to download, transform, & store many images from a remote web server
  - Part 3 of this case study focuses on the RxJava & Project Reactor implementations

```
Flux
   .fromIterable(getUrlList())
   .parallel()
   .runOn(Schedulers
              .boundedElastic())
   .map(...::downloadImage)
   .flatMap(...::transformImage)
   .map(...::storeImage)
   .sequential()
   .collectList()
   .block();
```

Applying Reactive Java Concurrency to Case Study ex4

Applying Reactive Java Concurrency to Case Study ex4



```java
    private static List<Future<Image>> transformImages
        (List<Future<Image>> downloadedImages) {
        // Create a new scope to execute virtual tasks, which exits
        // only after all tasks complete.
        try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
            // A List of Future<Image> objects that complete when the
            // images have been transformed asynchronously.
            var transformedImages = new ArrayList<Future<Image>>();

            // Iterate through the List of imageFutures.
            for (var imageFuture : downloadedImages) {
                transformedImages
                    // Append the transforming images at the end
                    // of the List.
                    .addAll( c: transformImage( executor: scope,
                                                 image: rethrowSupplier
                                                 ( function: imageFuture::get)
                                                 .get()));
            }

            rethrowRunnable( t: scope::join);
            // Scope doesn't exit until all concurrent tasks complete.
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Loom/ex4

End of Applying Java Structured Concurrency: Case Study ex4 (Part 3)