

Applying Java Structured Concurrency: Case Study ex4 (Part 1b)

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 classes used to program Java's structure concurrency model
- Case study ex4 evaluates the design & performance results of various Java concurrency models

```
try (var executor = Executors.  
    newVirtualThreadPerTaskExecutor())  
{  
    return urlList  
        .stream()  
  
        .map(url -> executor  
            .submit(() ->  
                loadImage(url)))  
  
        .toList();  
}
```

Learning Objectives in this Part of the Lesson

- Understand Java's structured concurrency model
- Recognize classes used to program Java's structured concurrency model
- Case study ex4 evaluates the design & performance results of various Java concurrency models
 - Part 1b of this case study focuses on the Java structured concurrency `Executors.newVirtualThreadPerTaskExecutor()`

```
try (var executor = Executors.  
    newVirtualThreadPerTaskExecutor())  
{  
    return urlList  
        .stream()  
  
        .map(url -> executor  
            .submit(() ->  
                downloadImage(url)))  
  
        .toList();  
}
```

Learning Objectives in this Part of the Lesson

- Understand Java's structured concurrency model
- Recognize classes used to program Java's structured concurrency model
- Case study ex4 evaluates the design & performance results of various Java concurrency models
- Part 1b of this case study focuses on the Java structured concurrency Executors.newVirtualThreadPerTaskExecutor()

```
try (var executor = Executors.  
    newVirtualThreadPerTaskExecutor())  
{  
    return urlList  
        .stream()  
  
        .map(url -> executor  
            .submit(() ->  
                downloadImage(url)))  
  
        .toList();  
}
```

The tasks in this case study are largely I/O-bound

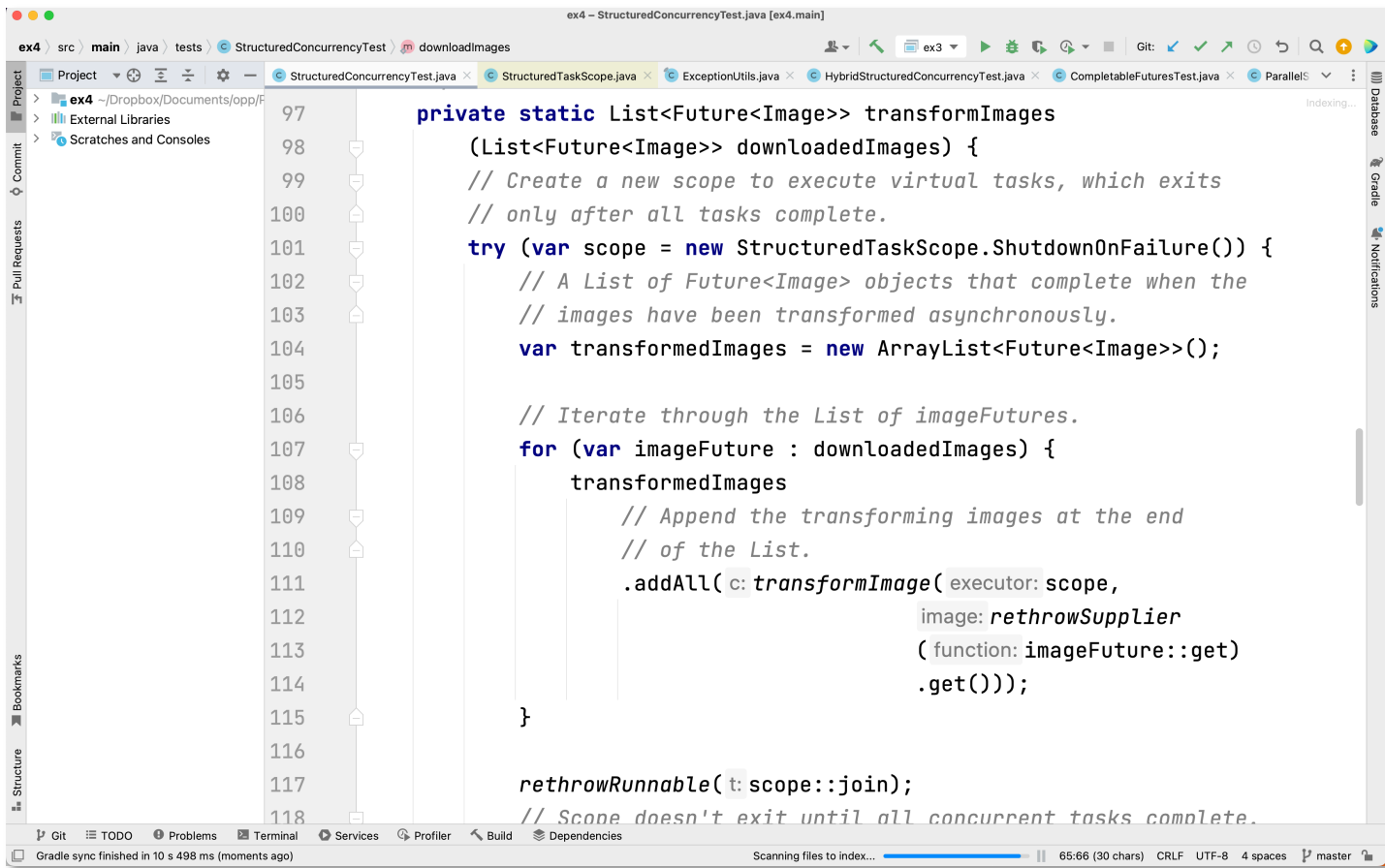
Learning Objectives in this Part of the Lesson

- Understand Java's structured concurrency model
- Recognize classes used to program Java's structured concurrency model
- Case study ex4 evaluates the design & performance results of various Java concurrency models
 - Part 1b of this case study focuses on the Java structured concurrency `Executors.newVirtualThreadPerTaskExecutor()`
 - This solution uses the Java sequential streams framework

```
try (var executor = Executors.  
    newVirtualThreadPerTaskExecutor())  
{  
    return urlList  
        .stream()  
  
        .map(url -> executor  
            .submit(() ->  
                downloadImage(url)))  
  
        .toList();  
}
```

Applying Java Structured Concurrency to Case Study ex4

Applying Java Structured Concurrency to Case Study ex4



The screenshot shows an IDE window titled "ex4 - StructuredConcurrencyTest.java [ex4.main]". The code is in Java and defines a method `transformImages` that takes a `List<Future<Image>>` and returns a `List<Future<Image>>`. The method uses `StructuredTaskScope` to manage a new scope for the transformation tasks. It iterates over the input list, adding each transformation task to the scope. The scope is configured to shutdown on failure. The method returns the transformed images list. The IDE interface includes a project explorer on the left, a toolbar at the top, and a status bar at the bottom.

```
97 private static List<Future<Image>> transformImages
98 (List<Future<Image>> downloadedImages) {
99     // Create a new scope to execute virtual tasks, which exits
100    // only after all tasks complete.
101    try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
102        // A List of Future<Image> objects that complete when the
103        // images have been transformed asynchronously.
104        var transformedImages = new ArrayList<Future<Image>>();
105
106        // Iterate through the List of imageFutures.
107        for (var imageFuture : downloadedImages) {
108            transformedImages
109                // Append the transforming images at the end
110                // of the List.
111                .addAll(c::transformImage(executor: scope,
112                                         image: rethrowSupplier
113                                         (function: imageFuture::get)
114                                         .get()));
115        }
116
117        rethrowRunnable(t: scope::join);
118        // Scope doesn't exit until all concurrent tasks complete.
```

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

End of Applying Java Structured Concurrency: Case Study ex4 (Part 1b)