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
- Case study ex4 evaluates the design & performance results of various Java concurrency models
- Part 3a of this case study focuses on the Project Reactor implementation

Flux
  .fromIterable(getUrlList())
  .parallel()
  .runOn(Schedulers.boundedElastic())
  .map(...::downloadImage)
  .flatMap(...::transformImage)
  .map(...::storeImage)
  .sequential()
  .collectList()
  .block();
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
• Case study ex4 evaluates the design & performance results of various Java concurrency models
• Part 3a of this case study focuses on the Project Reactor implementation

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

The tasks in this case study are largely I/O-bound
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
                .addAll(c:transformImage(
                    executor: scope,
                    image: rethrowSupplier
                        (function: imageFuture::get)
                        .get()));
        }

        rethrowRunnable(t: scope::join);
        // Scope doesn't exit until all concurrent tasks complete.
    }
}
```
End of Applying Java Structured Concurrency: Case Study ex4 (Part 3a)