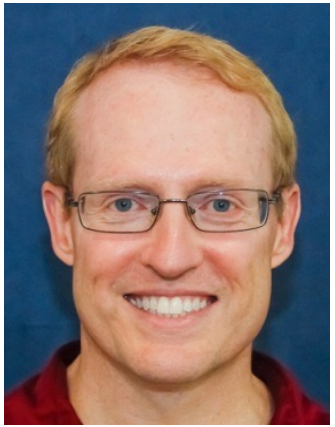
Applying Java Structured Concurrency: Case Study ex4 (Part 2)

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

```
Options.instance()
   .getUrlList()
   .parallelStream()
   .map(...::downloadImage)
   .map(...::transformImage)
   .reduce(Stream::concat)...
   .map(...::storeImage)
   .toList();
```

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 2 of this case study focuses on modern Java implementations that use the parallel streams & completable futures frameworks

```
Options.instance()
   .getUrlList()
   .parallelStream()
   .map(...::downloadImage)
   .map(...::transformImage)
   .reduce(Stream::concat)...
   .map(...::storeImage)
   .toList();
```

Applying Modern Java Concurrency to Case Study ex4

Applying Modern Java Concurrency to Case Study ex4

ex4 – ParallelStreamsTest.java [ex4.main]

```java
// Store the list of downloaded/tranformed images.
List<File> imageFiles = Options.instance() Options
    // Get the List of URLs.
    .getUrlList() List<URL>

    // Convert List into a parallel stream.
    .parallelStream() Stream<URL>

    // Transform URL to an image by downloading each image via
    // its URL.
    .map( mapper: FileAndNetUtils::downloadImage) Stream<Image>

    // Apply transforms to all images, yielding a stream of
    // stream of images.
    .map( mapper: ParallelStreamsTest::transformImage) Stream<Stream<Image>>

    // Convert the stream of stream of images into a stream
    // of images without using flatMap().
    .reduce( accumulator: Stream::concat).orElse( other: Stream.empty()) Stream

    // Store the images.
    .map( mapper: ParallelStreamsTest::storeImage) Stream<File>
```

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

End of Applying Java Structured Concurrency: Case Study ex4 (Part 2)