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

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
- Case study ex4 evaluates the design & performance results of various Java concurrency models
  - Part 2a of this case study focuses on modern Java implementations that use the parallel streams framework

Options.instance()
  .getUrlList()
  .parallelStream()
  .map(::{downloadImage})
  .map(::{transformImage})
  .reduce(Stream::concat)...
  .map(::{storeImage})
  .toList();
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 2a of this case study focuses on modern Java implementations that use the parallel streams framework

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

The tasks in this case study are largely I/O-bound
Applying Modern Java Concurrency to Case Study ex4
Applying Modern Java Concurrency to Case Study ex4

```java
// Store the list of downloaded/transformed images.
List<File> imageFiles = Options.instance().Options
    .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/douglasraigschmidt/LiveLessons/tree/master/Loom/ex4](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Loom/ex4)
End of Applying Java Structured Concurrency: Case Study ex4 (Part 2a)