

Applying Key Operators in the Flux Class: Case Study ex3 (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

- Part 2 of case study ex3 explores the use of Flux operators `filter()`, `generate()`, `flatMap()`, `fromIterable()`, `reduce()`, `collect()`, & the parallel thread pool to create, reduce, multiply, & display `BigFraction` objects asynchronously in a parallel thread pool

```
return Flux
    .generate(bigFractionEmitter)

    .take(sMAX_FRACTIONS)
    .flatMap(unreducedFraction ->
        reduceAndMultiplyFraction
            (unreducedFraction,
             Schedulers
               .parallel()))
    .collect(toList())

    .flatMap(list -> BigFractionUtils
        .sortAndPrintList(list,
                           sb));
```

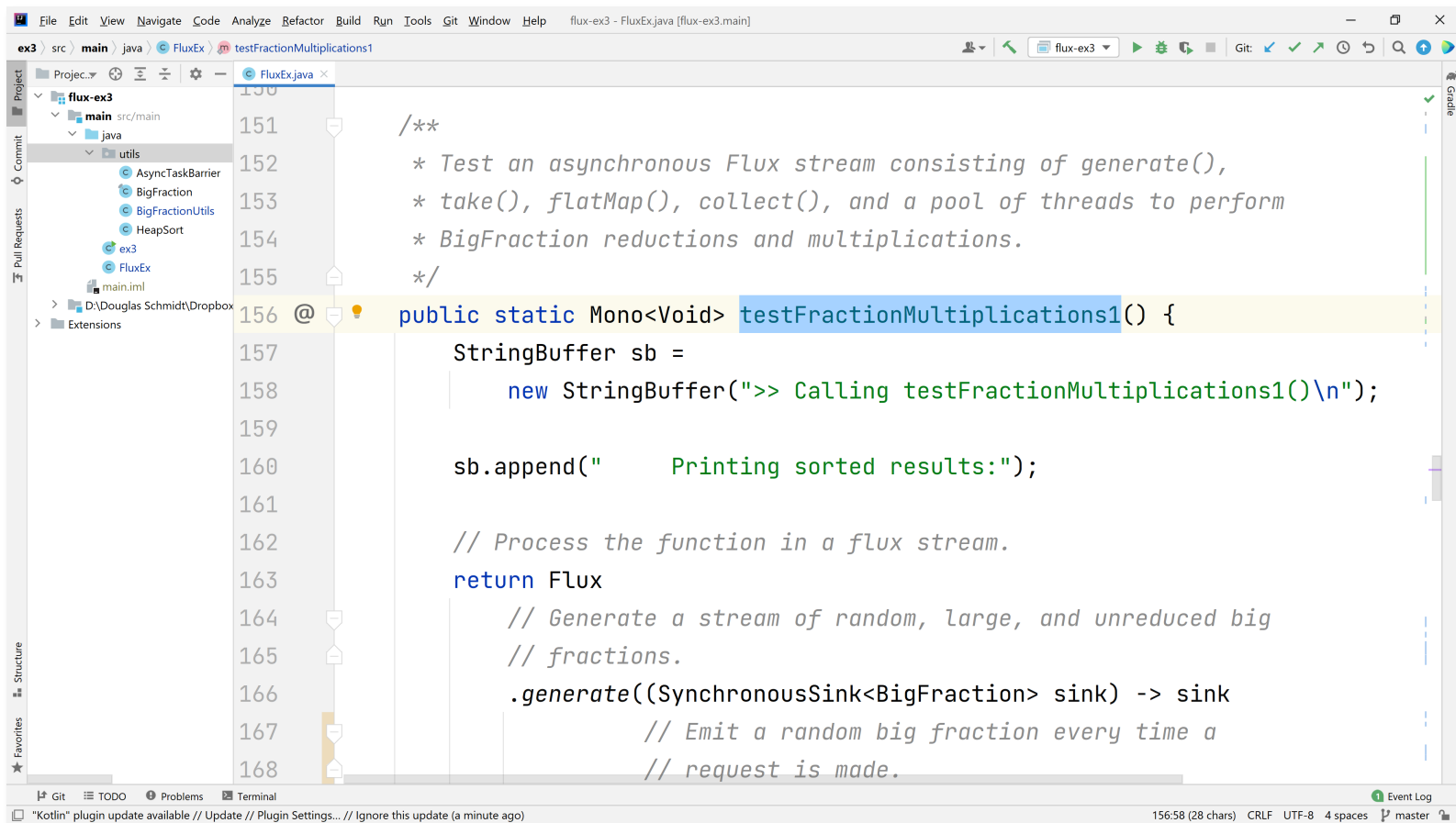
Learning Objectives in this Part of the Lesson

- Part 2 of case study ex3 explores the use of Flux operators `filter()`, `generate()`, `flatMap()`, `fromIterable()`, `reduce()`, `collect()`, & the parallel thread pool to create, reduce, multiply, & display `BigFraction` objects asynchronously in a parallel thread pool
- It also shows the use of `Mono` operators like `doOnNext()`, `map()`, `firstWithSignal()`, `subscribeOn()`, `flatMap()`, `fromCallable()`, & `then()`

```
return Mono
    .fromCallable(() -> BigFraction
        .reduce(unreducedFrac))
    .subscribeOn(scheduler)
    .doOnNext(result ->
        logBigFractionResult
            (unreducedFrac,
             sBigReducedFraction,
             result, sb))
    .map(reducedFraction ->
        reducedFraction
            .multiply
                (sBigReducedFraction));
```

Applying Key Operators in the Flux Class to ex3

Applying Key Operators in the Flux Class to ex3



The screenshot shows an IDE window titled "flux-ex3 - FluxEx.java [flux-ex3.main]". The left sidebar displays a project structure for "flux-ex3" with a "main" directory containing "java" and "utils" subdirectories. The "utils" directory contains several files, including "FluxEx". The main editor area shows the code for "testFractionMultiplications1" in "FluxEx.java". The code is as follows:

```
151 /**
152  * Test an asynchronous Flux stream consisting of generate(),
153  * take(), flatMap(), collect(), and a pool of threads to perform
154  * BigFraction reductions and multiplications.
155  */
156 @ public static Mono<Void> testFractionMultiplications1() {
157     StringBuffer sb =
158         new StringBuffer(">> Calling testFractionMultiplications1()\n");
159
160     sb.append("    Printing sorted results:");
161
162     // Process the function in a flux stream.
163     return Flux
164         // Generate a stream of random, large, and unreduced big
165         // fractions.
166         .generate((SynchronousSink<BigFraction> sink) -> sink
167             // Emit a random big fraction every time a
168             // request is made.
```

The bottom status bar shows "156:58 (28 chars) CRLF UTF-8 4 spaces master".

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex3

End of Applying Key Methods in the Flux Class: Case Study ex3 (Part 2)