

Applying Key Operators in the Parallel Flux Class: Case Study ex5 (Part 1)

Douglas C. Schmidt

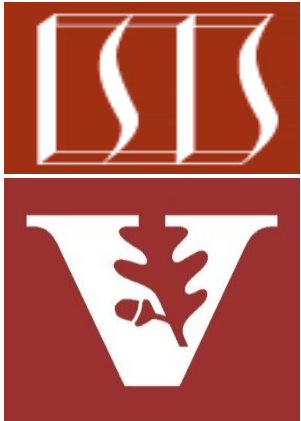
d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



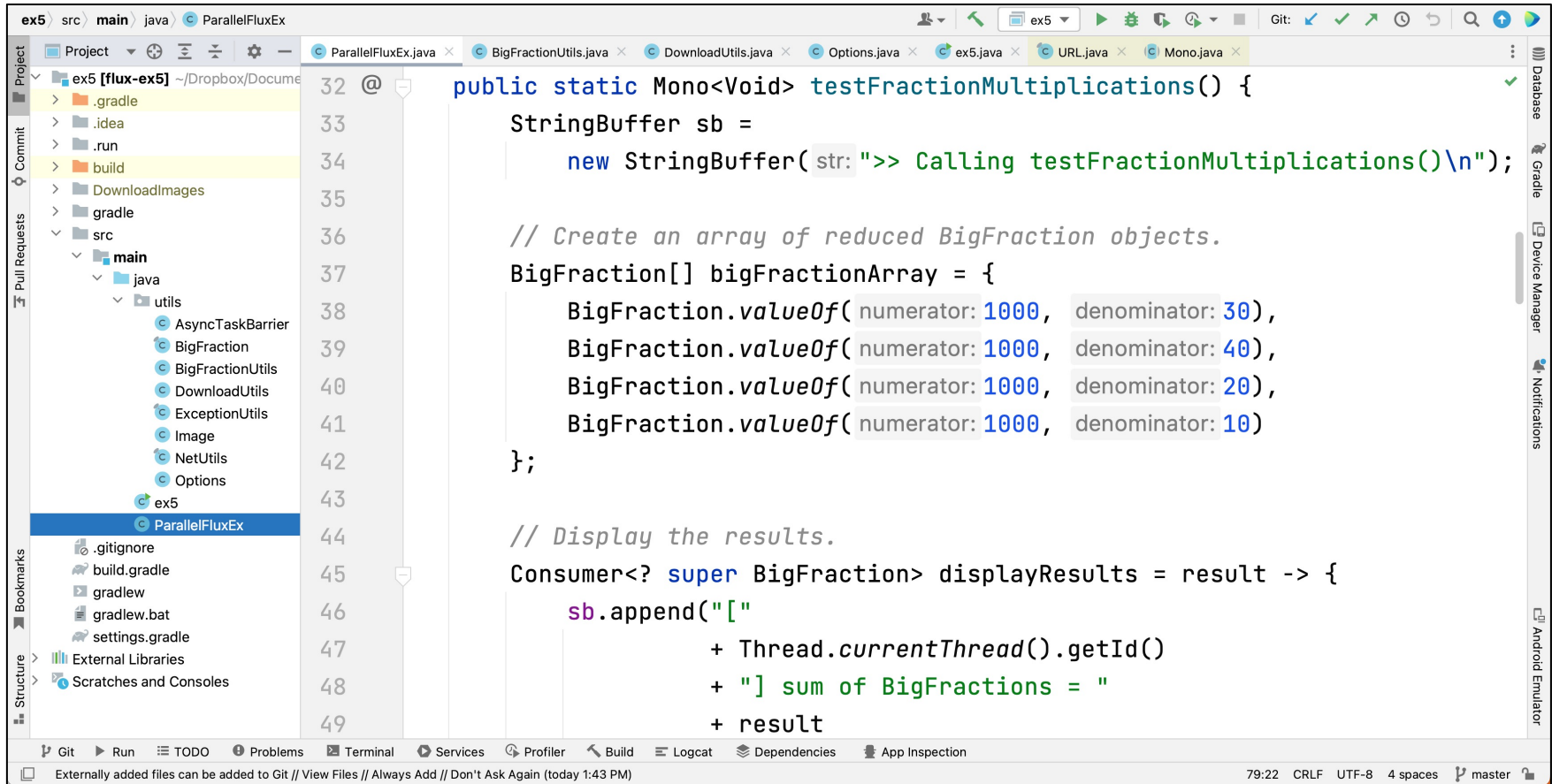
Learning Objectives in this Part of the Lesson

- Part 1 of case study ex5 shows how to multiply & add big fractions asynchronously & concurrently using Project Reactor operators, e.g.,
 - Flux operators `fromArray()`, `zipWith()`, `doOnNext()`, `doFinally()`, `then()`, & `parallel()`
 - ParallelFlux operators `runOn()`, `map()`, & `reduce()`
 - The `Schedulers.parallel()` thread pool

```
return Flux
    .fromArray(bigFractionArray)
    .parallel()
    .runOn
        (Schedulers.parallel())
    .map(bf -> bf
        .multiply(sBigReducedFrac))
    .reduce(BigFraction::add)
    .doOnSuccess(displayResults)
    .then();
```

Applying Key Operators in the ParallelFlux Class to ex5

Applying Key Operators in the ParallelFlux Class to ex5



```
32 @
33 public static Mono<Void> testFractionMultiplications() {
34     StringBuffer sb =
35         new StringBuffer(str: ">> Calling testFractionMultiplications()\n");
36
37     // Create an array of reduced BigFraction objects.
38     BigFraction[] bigFractionArray = {
39         BigFraction.valueOf(numerator: 1000, denominator: 30),
40         BigFraction.valueOf(numerator: 1000, denominator: 40),
41         BigFraction.valueOf(numerator: 1000, denominator: 20),
42         BigFraction.valueOf(numerator: 1000, denominator: 10)
43     };
44
45     // Display the results.
46     Consumer<? super BigFraction> displayResults = result -> {
47         sb.append("[ "
48             + Thread.currentThread().getId()
49             + " ] sum of BigFractions = "
50             + result
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex5

End of Applying Key
Operators in the ParallelFlux
Class: Case Study ex5
(Part 2)