Applying Key Operators in Project Reactor: Case Study ex4 (Part 3)

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 3 of case study ex4 applies Flux operators flatMap() & subscribe() to create, multiply, & display BigFraction objects asynchronously

```
Mono
   .fromSupplier(() ->
      makeBigFraction
         (sRANDOM, true))

   .repeat(sMAX_FRACTIONS - 1)

   .flatMap(bf1 ->
      multiplyFraction(bf1,
          sBigReducedFraction,
          Schedulers.parallel(),
          sb))

   .subscribe
      (backpressureSubscriber);
```

Learning Objectives in this Part of the Lesson

- Part 3 of case study ex4 applies Flux operators flatMap() & subscribe() to create, multiply, & display BigFraction objects asynchronously

```
Mono
   .fromSupplier(() ->
      makeBigFraction
        (sRANDOM, true))

   .repeat(sMAX_FRACTIONS - 1)

   .flatMap(bf1 ->
      multiplyFraction(bf1,
          sBigReducedFraction,
          Schedulers.parallel(),
          sb))

   .subscribe
      (backpressureSubscriber);
```

This example *does* apply backpressure via the registered subscriber

Learning Objectives in this Part of the Lesson

- Part 3 of case study ex4 applies Flux operators flatMap() & subscribe() to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators fromSupplier(), repeat(), & subscribeOn()

```
Mono
   .fromSupplier(() ->
      makeBigFraction
        (sRANDOM, true))

   .repeat(sMAX_FRACTIONS - 1)

   .flatMap(bf1 ->
      multiplyFraction(bf1,
           sBigReducedFraction,
           Schedulers.parallel(),
           sb))

   .subscribe
      (backpressureSubscriber);
```

Learning Objectives in this Part of the Lesson

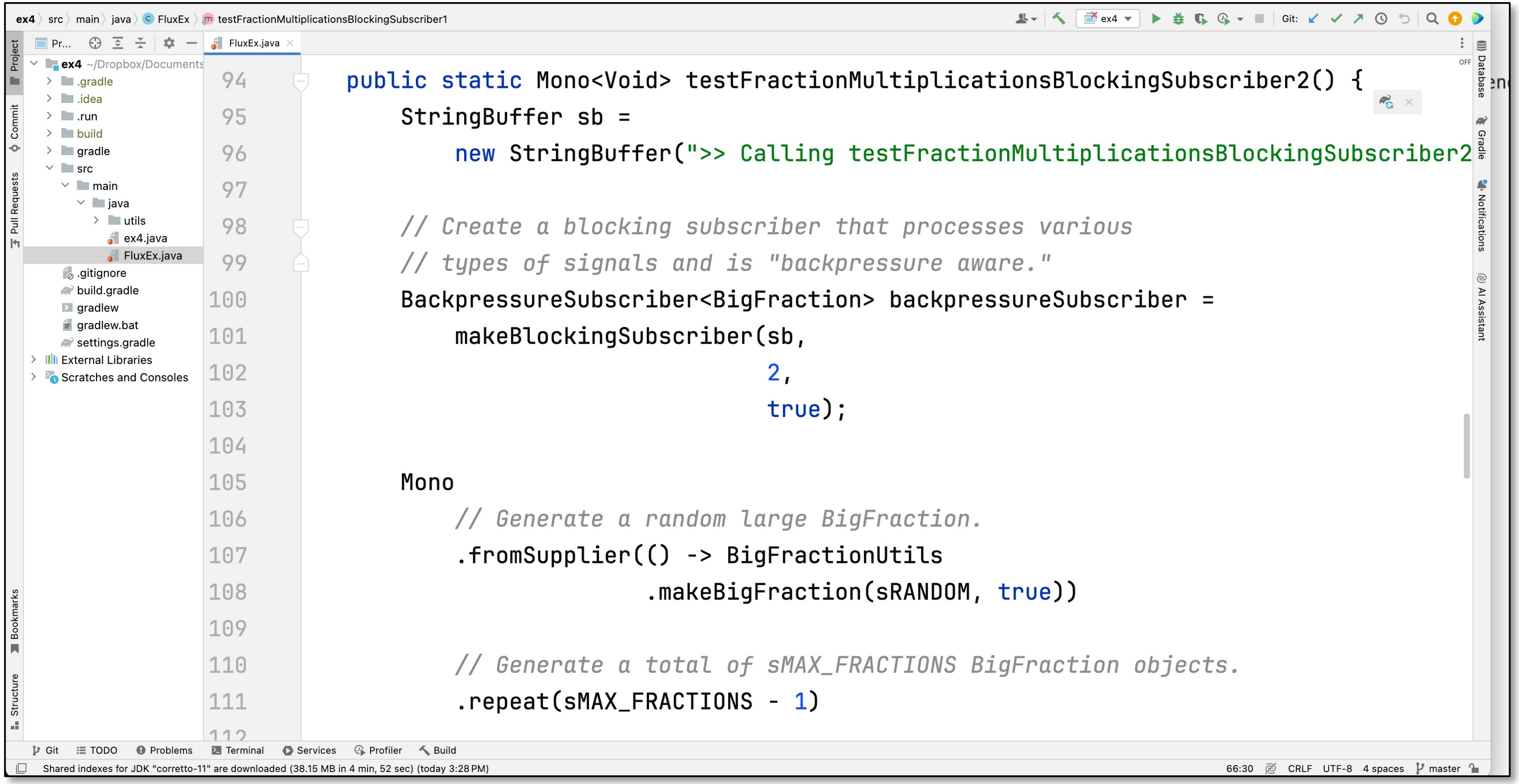
- Part 3 of case study ex4 applies Flux operators flatMap() & subscribe() to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators fromSupplier(), repeat(), & subscribeOn()
  - In addition, it shows how to use the generic blocking Subscriber
    - This subscriber is "backpressure aware"

```
class BackpressureSubscriber<T>
   implements CoreSubscriber<T> {
   ...
   @Override
   public void onSubscribe
    (Subscription subscription){
     mSubscription =
       subscription;

     subscription
       .request(mRequestSize);
   }
   ...
}
```

Applying Key Operators in Project Reactor to ex4

Applying Key Operators in Project Reactor to ex4



```java
public static Mono<Void> testFractionMultiplicationsBlockingSubscriber2() {
    StringBuffer sb =
        new StringBuffer(">> Calling testFractionMultiplicationsBlockingSubscriber2

    // Create a blocking subscriber that processes various
    // types of signals and is "backpressure aware."
    BackpressureSubscriber<BigFraction> backpressureSubscriber =
        makeBlockingSubscriber(sb,
                               2,
                               true);

    Mono
        // Generate a random large BigFraction.
        .fromSupplier(() -> BigFractionUtils
                      .makeBigFraction(sRANDOM, true))

        // Generate a total of sMAX_FRACTIONS BigFraction objects.
        .repeat(sMAX_FRACTIONS - 1)
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4

End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 3)