Applying Key Operators in Project Reactor: Case Study ex4 (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 ex4 applies Flux operators subscribe(), flatMap(), & fromArray() to create, multiply, & display BigFraction objects asynchronously

```
Mono
   .fromSupplier(() ->
           .makeBigFraction
               (sRANDOM, true))
   .flatMapMany(bf1 -> Flux
      .fromArray(bigFractionArray)
      .flatMapMany(bf2 -> Mono
           .fromCallable(() -> bf2)
           .subscribeOn(scheduler)
           .map(___ -> bf2
                .multiply(bf1))))

   .subscribe(blockingSubscriber);
```

This example does not apply backpressure at all

Learning Objectives in this Part of the Lesson

- Part 1 of case study ex4 applies Flux operators subscribe(), flatMap(), & fromArray() to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators fromSupplier(), map(), flatMapMany(), & subscribeOn()

```
Mono
   .fromSupplier(() ->
           .makeBigFraction
               (sRANDOM, true))
   .flatMapMany(bf1 -> Flux
      .fromArray(bigFractionArray)
      .flatMapMap(bf2 -> Mono
           .fromCallable(() -> bf2)
           .subscribeOn(scheduler)
           .map(___ -> bf2
                .multiply(bf1))))

   .subscribe(blockingSubscriber);
```

Learning Objectives in this Part of the Lesson

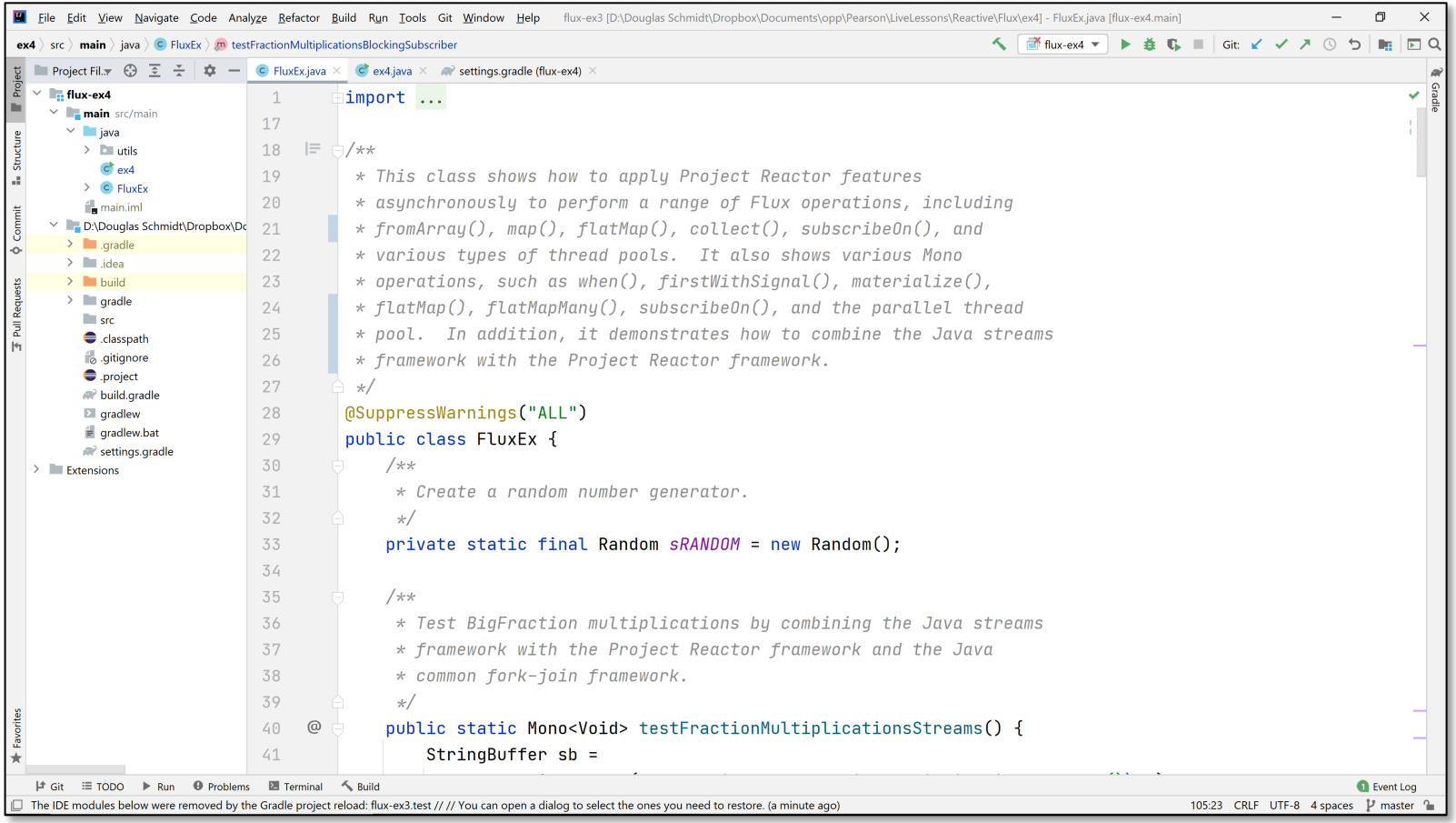
- Part 1 of case study ex4 applies Flux operators subscribe(), flatMap(), & fromArray() to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators fromSupplier(), map(), flatMapMany(), & subscribeOn()
  - In addition, it shows how to create & use a generic blocking Subscriber
    - Can be applied to workaround the lack of a blockingSubscribe() operator

```
class BlockingSubscriber<T>
      implements Subscriber<T> {
  ...
  final CountDownLatch mLatch;
  ...
  @Override
  public void onComplete() {
    ...
    mLatch.countDown();
  }
  ...
}
```

In contrast, RxJava has a blockingSubscribe() operator on Observable

Applying Key Operators in Project Reactor to ex4

Applying Key Operators in Project Reactor to ex4



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

End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 1)