

# Applying Key Operators in Project Reactor: Case Study ex4 (Part 4)

**Douglas C. Schmidt**

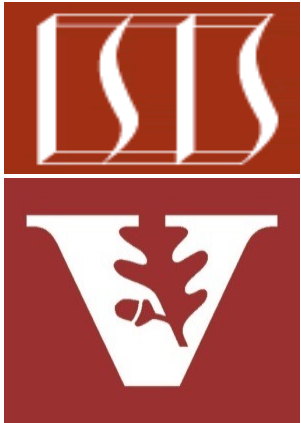
**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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 4 of case study ex4 applies Flux operators `create()`, `flatMap()`, & `subscribe()`, as well as `FluxSink` to create, multiply, & display `BigFraction` objects asynchronously

## Flux

```
.create (makeEmitter (count ,  
                                sb) ,
```

```
    FluxSink
```

```
    .OverflowStrategy  
    .ERROR)
```

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

```
.subscribe  
  (blockingSubscriber) ;
```

---

This subscriber is "backpressure-unaware"

# Learning Objectives in this Part of the Lesson

---

- Part 4 of case study ex4 applies Flux operators create(), flatMap(), & subscribe(), as well as FluxSink to create, multiply, & display BigFraction objects asynchronously

**Flux**

```
.create (makeEmitter (count ,  
sb) ,
```

**FluxSink**

```
.OverflowStrategy  
.ERROR)
```

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

```
.subscribe  
(blockingSubscriber) ;
```

---

This example applies an overflow strategy

# Learning Objectives in this Part of the Lesson

---

- Part 4 of case study ex4 applies Flux operators `create()`, `flatMap()`, & `subscribe()`, as well as FluxSink to `create`, `multiply`, & `display` `BigFraction` objects asynchronously
- It also shows how to use `Mono` operators `fromSupplier()` & `subscribeOn()`

```
Mono<BigFraction>
multiplyFraction(BigFraction bf1,
                BigFraction bf2,
                Scheduler scheduler,
                StringBuffer sb) {
    return Mono
        .fromSupplier(() -> bf1
                    .multiply(bf2))
        .subscribeOn(scheduler);
}
```

# Learning Objectives in this Part of the Lesson

---

- Part 4 of case study ex4 applies Flux operators `create()`, `flatMap()`, & `subscribe()`, as well as FluxSink to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators `fromSupplier()` & `subscribeOn()`
  - In addition, it shows how to use a generic blocking Subscriber

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

# Learning Objectives in this Part of the Lesson

---

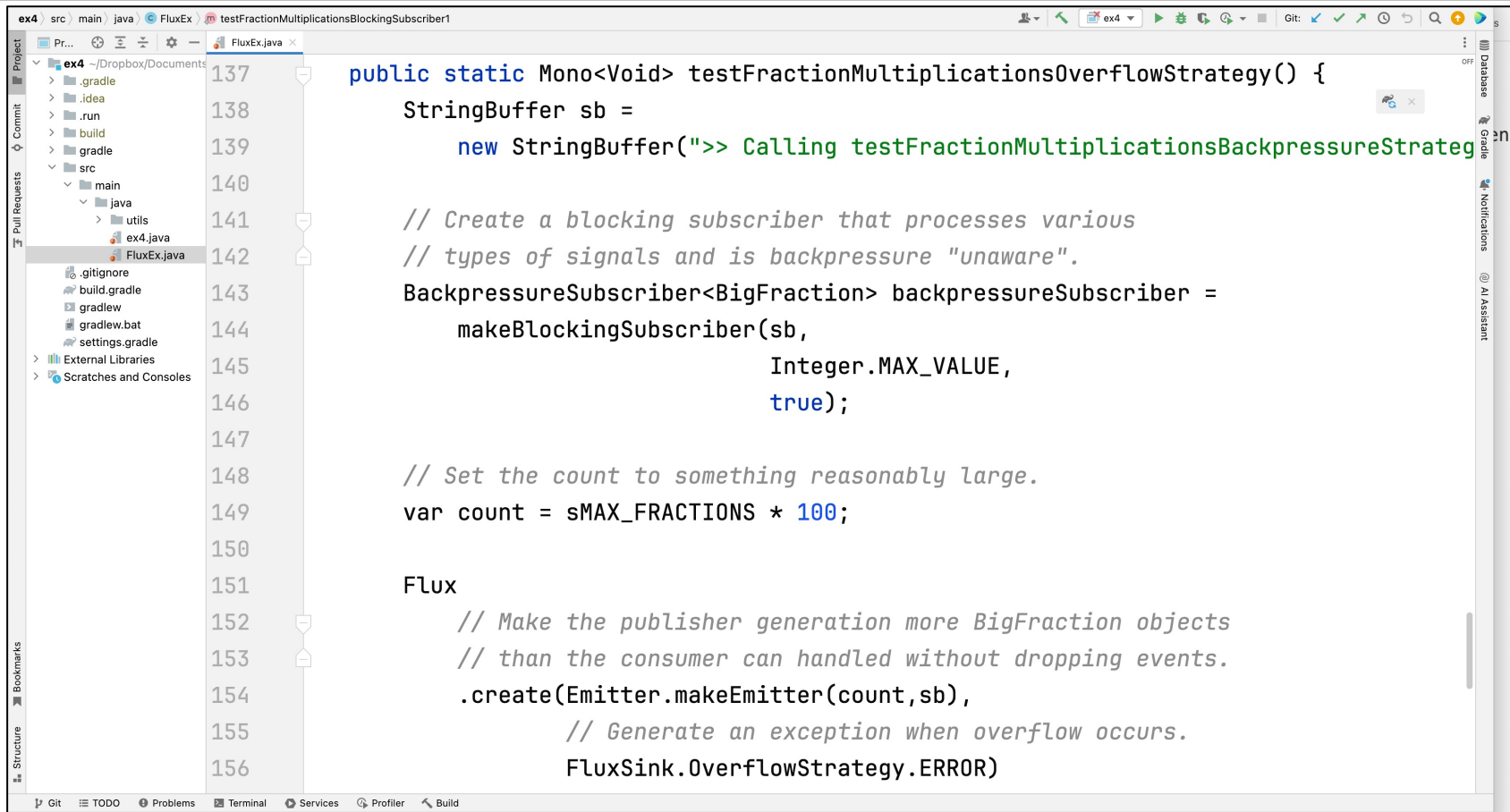
- Part 4 of case study ex4 applies Flux operators `create()`, `flatMap()`, & `subscribe()`, as well as FluxSink to create, multiply, & display BigFraction objects asynchronously
  - It also shows how to use Mono operators `fromSupplier()` & `subscribeOn()`
  - In addition, it shows how to use a generic blocking Subscriber  
`return blockingSubscriber  
    .await ();`

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

---

# Applying Key Operators in Project Reactor to ex4

# Applying Key Operators in Project Reactor to ex4



```
137 public static Mono<Void> testFractionMultiplicationsOverflowStrategy() {
138     StringBuffer sb =
139         new StringBuffer(">> Calling testFractionMultiplicationsBackpressureStrategy");
140
141     // Create a blocking subscriber that processes various
142     // types of signals and is backpressure "unaware".
143     BackpressureSubscriber<BigFraction> backpressureSubscriber =
144         makeBlockingSubscriber(sb,
145             Integer.MAX_VALUE,
146             true);
147
148     // Set the count to something reasonably large.
149     var count = sMAX_FRACTIONS * 100;
150
151     Flux
152         // Make the publisher generation more BigFraction objects
153         // than the consumer can handled without dropping events.
154         .create(Emitter.makeEmitter(count, sb),
155             // Generate an exception when overflow occurs.
156             FluxSink.OverflowStrategy.ERROR)
```

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4)



---

# End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 4)