

Overview of the BigFraction Case Studies

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

- Understand key classes in the RxJava API
- Be aware of the structure & functionality of the BigFraction case studies
 - These case studies showcase many operators in the RxJava Single, Observable, & Flowable classes

<code><<Java Class>></code> BigFraction
<ul style="list-style-type: none"><code>mNumerator: BigInteger</code><code>mDenominator: BigInteger</code>
<ul style="list-style-type: none"><code>BigFraction()</code><code>valueOf(Number):BigFraction</code><code>valueOf(Number,Number):BigFraction</code><code>valueOf(String):BigFraction</code><code>valueOf(Number,Number,boolean):BigFraction</code><code>reduce(BigFraction):BigFraction</code><code>getNumerator():BigInteger</code><code>getDenominator():BigInteger</code><code>add(Number):BigFraction</code><code>subtract(Number):BigFraction</code><code>multiply(Number):BigFraction</code><code>divide(Number):BigFraction</code><code>gcd(Number):BigFraction</code><code>toMixedString():String</code>

Overview of the BigFraction Class

Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class

<p><<Java Class>> BigFraction</p>
<p>F mNumerator: BigInteger F mDenominator: BigInteger</p>
<p>C BigFraction() S valueOf(Number):BigFraction S valueOf(Number,Number):BigFraction S valueOf(String):BigFraction S valueOf(Number,Number,boolean):BigFraction S reduce(BigFraction):BigFraction F getNumerator():BigInteger F getDenominator():BigInteger S add(Number):BigFraction S subtract(Number):BigFraction S multiply(Number):BigFraction S divide(Number):BigFraction S gcd(Number):BigFraction S toMixedString():String</p>

See LiveLessons/blob/master/Java8/ex8/src/utils/BigFraction.java

Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
- Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator

<<Java Class>>	
🔍 BigFraction	
📄 F	mNumerator: BigInteger
📄 F	mDenominator: BigInteger
📄 C	BigFraction()
📄 S	valueOf(Number):BigFraction
📄 S	valueOf(Number,Number):BigFraction
📄 S	valueOf(String):BigFraction
📄 S	valueOf(Number,Number,boolean):BigFraction
📄 S	reduce(BigFraction):BigFraction
📄 F	getNumerator():BigInteger
📄 F	getDenominator():BigInteger
📄 S	add(Number):BigFraction
📄 S	subtract(Number):BigFraction
📄 S	multiply(Number):BigFraction
📄 S	divide(Number):BigFraction
📄 S	gcd(Number):BigFraction
📄 S	toMixedString():String

See docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html

Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
 - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator
- Factory methods to “reduce” fractions
 - $44/55 \rightarrow 4/5$
 - $12/24 \rightarrow 1/2$
 - $144/216 \rightarrow 2/3$

<<Java Class>>	
BigFraction	
F	mNumerator: BigInteger
F	mDenominator: BigInteger
C	BigFraction()
S	valueOf(Number):BigFraction
S	valueOf(Number,Number):BigFraction
S	valueOf(String):BigFraction
F	valueOf(Number,Number,boolean):BigFraction
S	reduce(BigFraction):BigFraction
F	getNumerator():BigInteger
F	getDenominator():BigInteger
C	add(Number):BigFraction
C	subtract(Number):BigFraction
C	multiply(Number):BigFraction
C	divide(Number):BigFraction
C	gcd(Number):BigFraction
C	toMixedString():String


















Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
 - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator
 - Factory methods to “reduce” fractions
 - Factory methods to create “non-reduced” fractions (& then reduce them)
 - e.g., 12/24 (\rightarrow 1/2)

<<Java Class>>	
BigFraction	
F mNumerator: BigInteger	
F mDenominator: BigInteger	
C BigFraction()	
S <u>valueOf(Number):BigFraction</u>	
S <u>valueOf(Number,Number):BigFraction</u>	
S <u>valueOf(String):BigFraction</u>	
S <u>valueOf(Number,Number,boolean):BigFraction</u>	
S <u>reduce(BigFraction):BigFraction</u>	
F getNumerator():BigInteger	
F getDenominator():BigInteger	
C add(Number):BigFraction	
C subtract(Number):BigFraction	
C multiply(Number):BigFraction	
C divide(Number):BigFraction	
C gcd(Number):BigFraction	
C toMixedString():String	

Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
 - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator
 - Factory methods to “reduce” fractions
 - Factory methods to create “non-reduced” fractions (& then reduce them)
- Arbitrary-precision fraction arithmetic
 - e.g., $18/4 \times 2/3 = 3$

<<Java Class>>	
	BigFraction
	mNumerator: BigInteger
	mDenominator: BigInteger
	BigFraction()
	valueOf(Number):BigFraction
	valueOf(Number,Number):BigFraction
	valueOf(String):BigFraction
	valueOf(Number,Number,boolean):BigFraction
	reduce(BigFraction):BigFraction
	getNumerator():BigInteger
	getDenominator():BigInteger
	add(Number):BigFraction
	subtract(Number):BigFraction
	multiply(Number):BigFraction
	divide(Number):BigFraction
	gcd(Number):BigFraction
	toMixedString():String

Overview of the BigFraction Class

- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
 - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator
 - Factory methods to “reduce” fractions
 - Factory methods to create “non-reduced” fractions (& then reduce them)
 - Arbitrary-precision fraction arithmetic
 - Create a mixed fraction from an improper fraction
 - e.g., $18/4 \rightarrow 4 \frac{1}{2}$

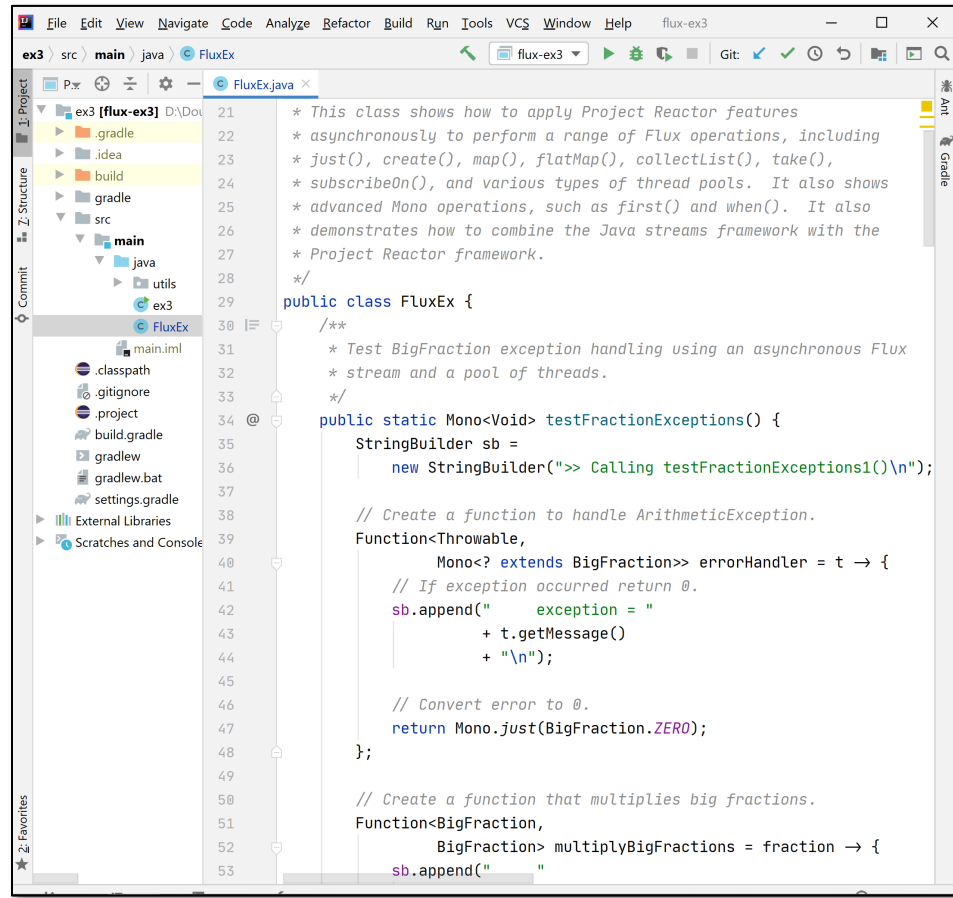
<<Java Class>>	
BigFraction	
F	mNumerator: BigInteger
F	mDenominator: BigInteger
C	BigFraction()
S	valueOf(Number):BigFraction
S	valueOf(Number,Number):BigFraction
S	valueOf(String):BigFraction
S	valueOf(Number,Number,boolean):BigFraction
S	reduce(BigFraction):BigFraction
F	getNumerator():BigInteger
F	getDenominator():BigInteger
	add(Number):BigFraction
	subtract(Number):BigFraction
	multiply(Number):BigFraction
	divide(Number):BigFraction
	gcd(Number):BigFraction
	toMixedString():String

See www.mathsisfun.com/improper-fractions.html

Overview of the BigFraction Case Studies

Overview of the BigFraction Case Studies

- These case studies show how to create, reduce, multiply, & display BigFraction objects synchronously, asynchronously, & concurrently using RxJava framework features



```
21  * This class shows how to apply Project Reactor features
22  * asynchronously to perform a range of FLUX operations, including
23  * just(), create(), map(), flatMap(), collectList(), take(),
24  * subscribeOn(), and various types of thread pools. It also shows
25  * advanced Mono operations, such as first() and when(). It also
26  * demonstrates how to combine the Java streams framework with the
27  * Project Reactor framework.
28  */
29
30  public class FluxEx {
31      /**
32       * Test BigFraction exception handling using an asynchronous Flux
33       * stream and a pool of threads.
34       */
35      @Test
36      public static Mono<Void> testFractionExceptions() {
37          StringBuilder sb =
38              new StringBuilder(">> Calling testFractionExceptions1()\n");
39
40          // Create a function to handle ArithmeticException.
41          Function<Throwable,
42              Mono<? extends BigFraction>> errorHandler = t -> {
43              // If exception occurred return 0.
44              sb.append("    exception = "
45                  + t.getMessage()
46                  + "\n");
47
48              // Convert error to 0.
49              return Mono.just(BigFraction.ZERO);
50          };
51
52          // Create a function that multiplies big fractions.
53          Function<BigFraction,
54              BigFraction> multiplyBigFractions = fraction -> {
55              sb.append("    ")
```

Overview of the BigFraction Case Studies

- The RxJava Single case studies show how to create, reduce, multiply, & display BigFraction objects using many Single features
- e.g., fromCallable(), zipWith(), zipArray(), doOnSuccess(), map(), ignoreElement(), subscribeOn(), ambArray(), & the parallel thread pool

```
BigFraction unreducedFraction =
    makeBigFraction(...);

return Single
    .fromCallable(() -> BigFraction
        .reduce(unreducedFraction))
    .subscribeOn
        (Schedulers.single())
    .map(result ->
        result.toMixedString())
    .doOnSuccess(result ->
        System.out.println
            ("big fraction = "
            + result + "\n"))
    .ignoreElement();
```

Overview of the BigFraction Case Studies

- The RxJava Observable case studies show how to create, reduce, multiply, & display Big Fraction objects using many Observable features
- e.g., `fromCallable()`, `map()`, `create()`, `interval()`, `filter()`, `doOnNext()`, `blockingSubscribe()`, `take()`, `doOnComplete()`, `subscribe()`, `flatMap()`, `fromIterable()`, `subscribeOn()`, `observeOn()`, `range()`, `count()`, `collect()`, & various thread pools

```
return Observable
    .fromArray(bigFractions)

    .subscribeOn(scheduler)

    .flatMap(reducedFraction ->
        Observable
            .fromCallable(() ->
                reducedFraction.multiply
                    (sBigReducedFraction))

        .subscribeOn
            (scheduler))

    .reduce(BigFraction::add);
```

Overview of the BigFraction Case Studies

- The RxJava Flowable case studies show how to create, reduce, multiply, & display Big Fraction objects using Flowable & ParallelFlowable features
- e.g., `fromArray()`, `parallel()`, `runOn()`, `flatMap()`, `reduce()`, `sequential()`, & the `Scheduler.computation()` thread pool

```
return Flowable
    .fromArray(bigFractions)

    .parallel()

    .runOn(Scheduler.computation())

    .flatMap(bigFraction ->
        bigFraction.multiply
            (sBigReducedFraction))

    .sequential()

    .reduce(BigFraction::add)

    ...
```

See github.com/douglascraigshmidt/LiveLessons/tree/master/Reactive/Flowable

End of Overview of the BigFraction Case Studies