

Overview of the BigFraction Case Studies

Douglas C. Schmidt

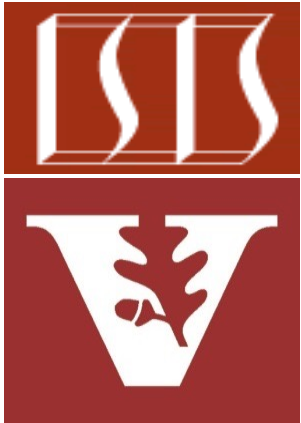
d.schmidt@vanderbilt.edu

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

<<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

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

<<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

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>>	
G 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 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
 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

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>>	
G 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

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()	
 <u>valueOf(Number):BigFraction</u>	
 <u>valueOf(Number,Number):BigFraction</u>	
 <u>valueOf(String):BigFraction</u>	
 <u>valueOf(Number,Number,boolean):BigFraction</u>	
 <u>reduce(BigFraction):BigFraction</u>	
 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**

 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

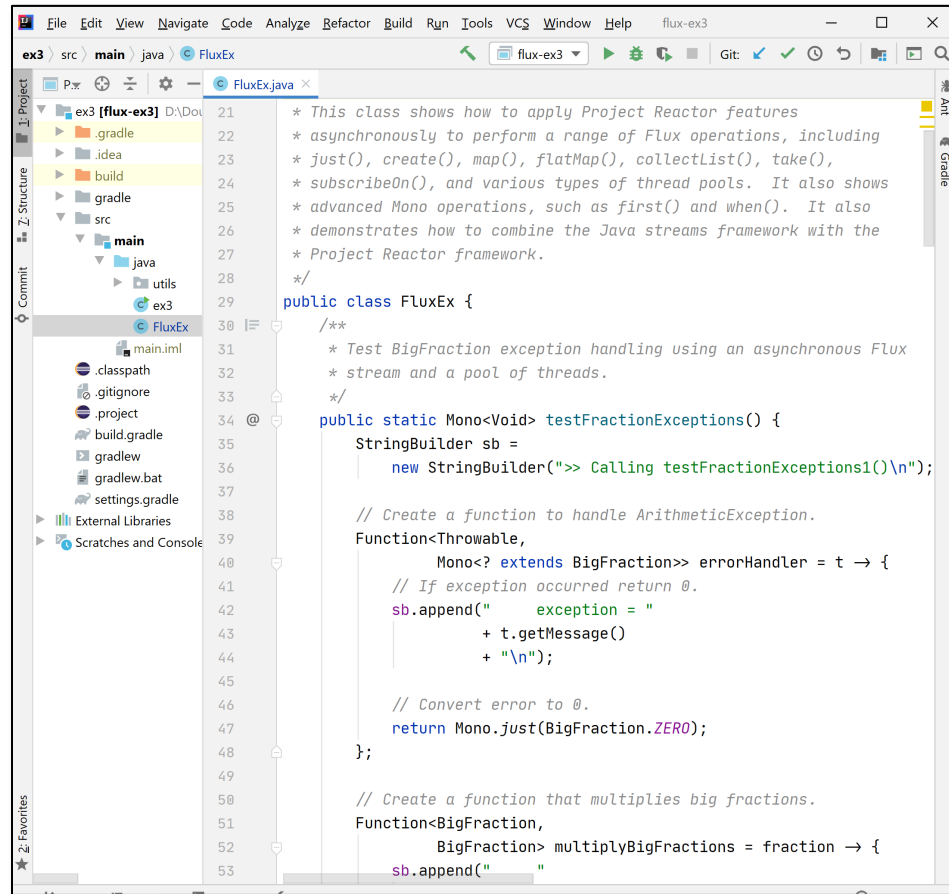
 **toMixedString():String**

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



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help flux-ex3
ex3 > src > main > java > FluxEx
FluxEx.java
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 public class FluxEx {
30     /**
31     * Test BigFraction exception handling using an asynchronous Flux
32     * stream and a pool of threads.
33     */
34     @Test
35     public static Mono<Void> testFractionExceptions() {
36         StringBuilder sb =
37             new StringBuilder(">> Calling testFractionExceptions1()\n");
38
39         // Create a function to handle ArithmeticException.
40         Function<Throwable,
41             Mono<? extends BigFraction>> errorHandler = t -> {
42             // If exception occurred return 0.
43             sb.append("    exception = "
44                 + t.getMessage()
45                 + "\n");
46
47             // Convert error to 0.
48             return Mono.just(BigFraction.ZERO);
49         };
50
51         // Create a function that multiplies big fractions.
52         Function<BigFraction,
53             BigFraction> multiplyBigFractions = fraction -> {
54             sb.append("    "
55                 + fraction.toString()
56                 + "\n");
57             return fraction.multiply(fraction);
58         };
59     }
60 }
```

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();
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Single

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(bigFractionList)

    .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(bigFractionList)

    .parallel()

    .runOn(Scheduler.computation())

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

    .sequential()

    .reduce(BigFraction::add)

    ...
```

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

End of Overview of the BigFraction Case Studies