Overview of the BigFraction Case Studies

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

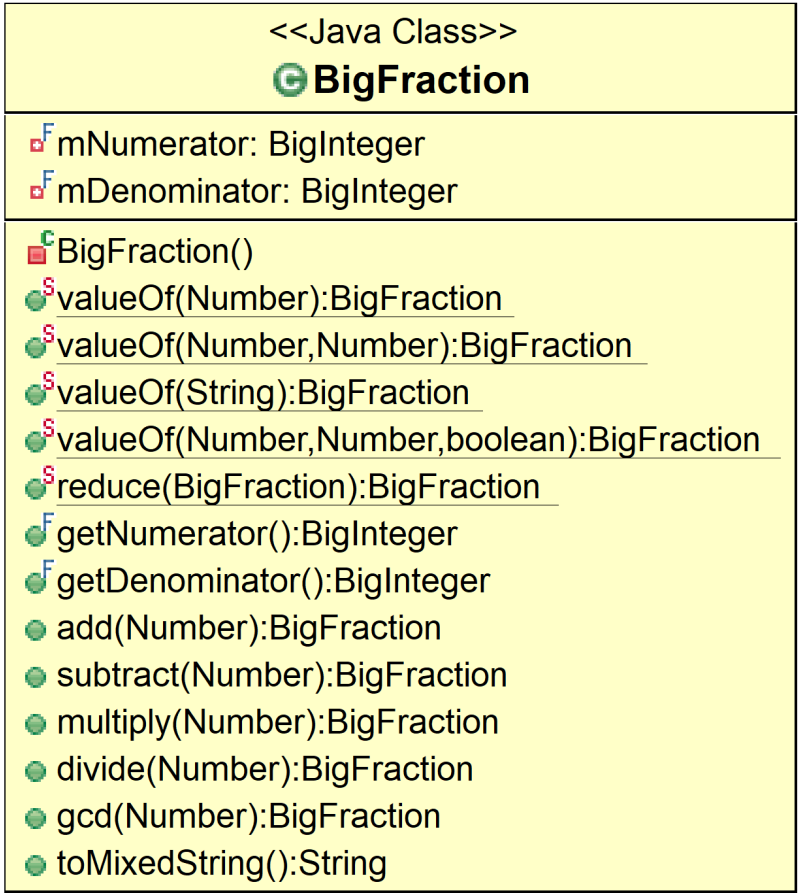
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

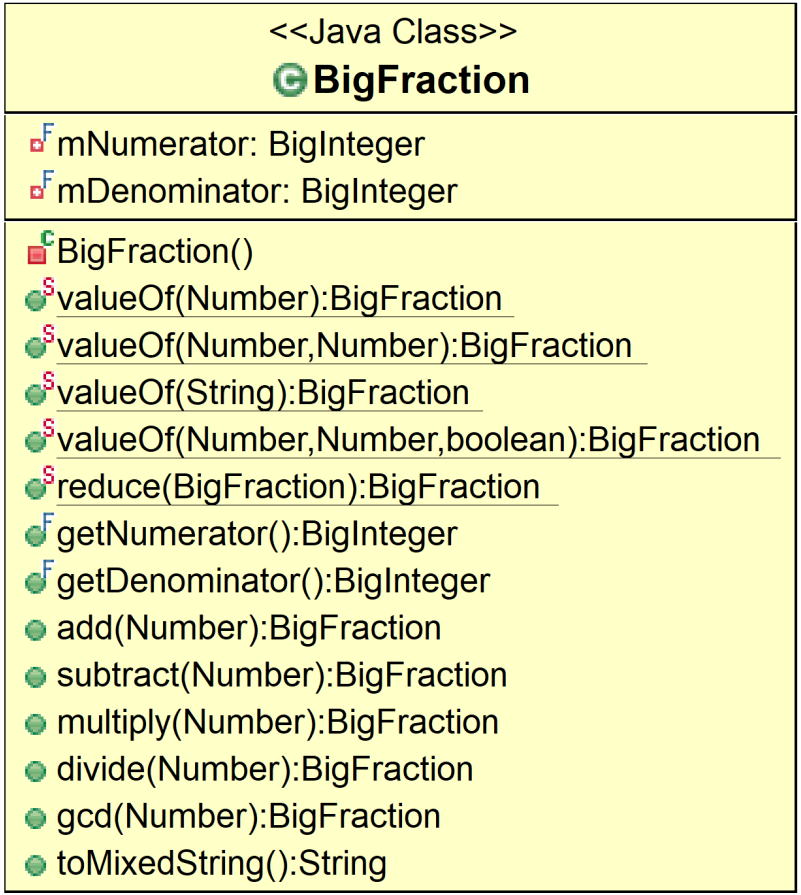
- Understand key classes in the RxJava API
- Be aware of the structure & functionality of the BigFraction case studies
  - These case studies showcase many operators in the RxJava Single, Observable, & Flowable classes



Overview of the BigFraction Class

Overview of the BigFraction Class

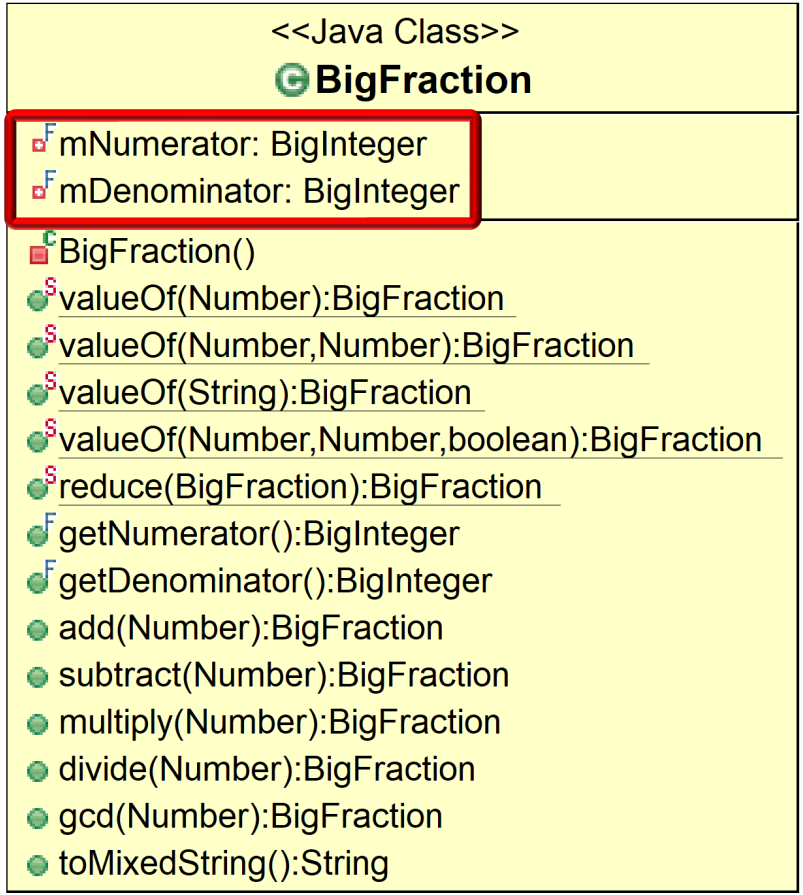
- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class



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

Overview of the BigFraction Class

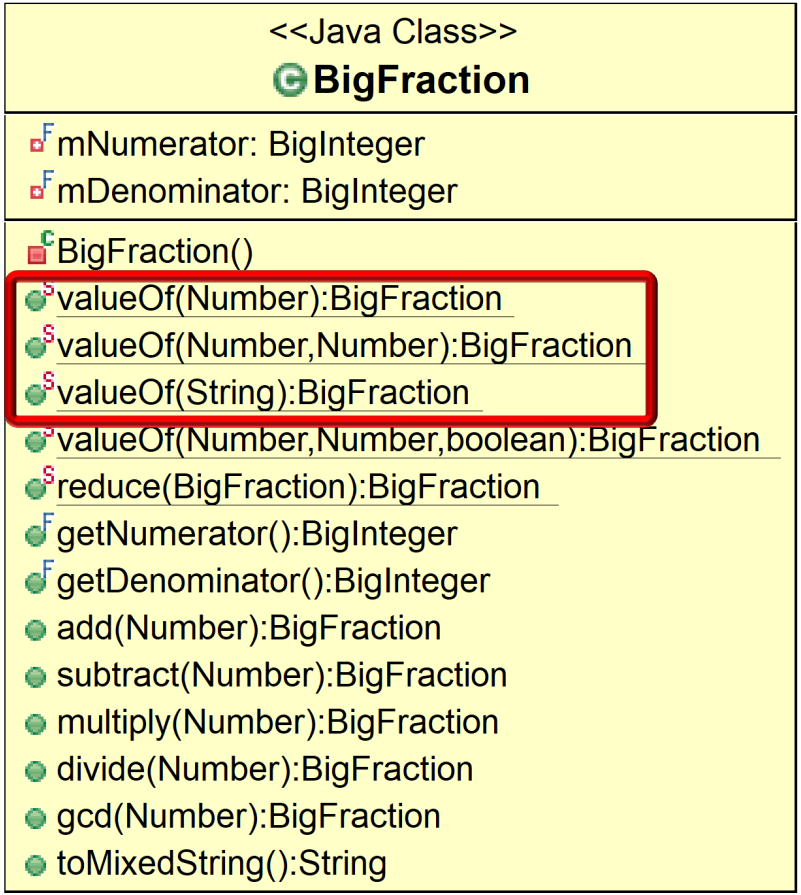
- Upcoming lessons show how to apply RxJava features in the context of a BigFraction class
  - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator



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

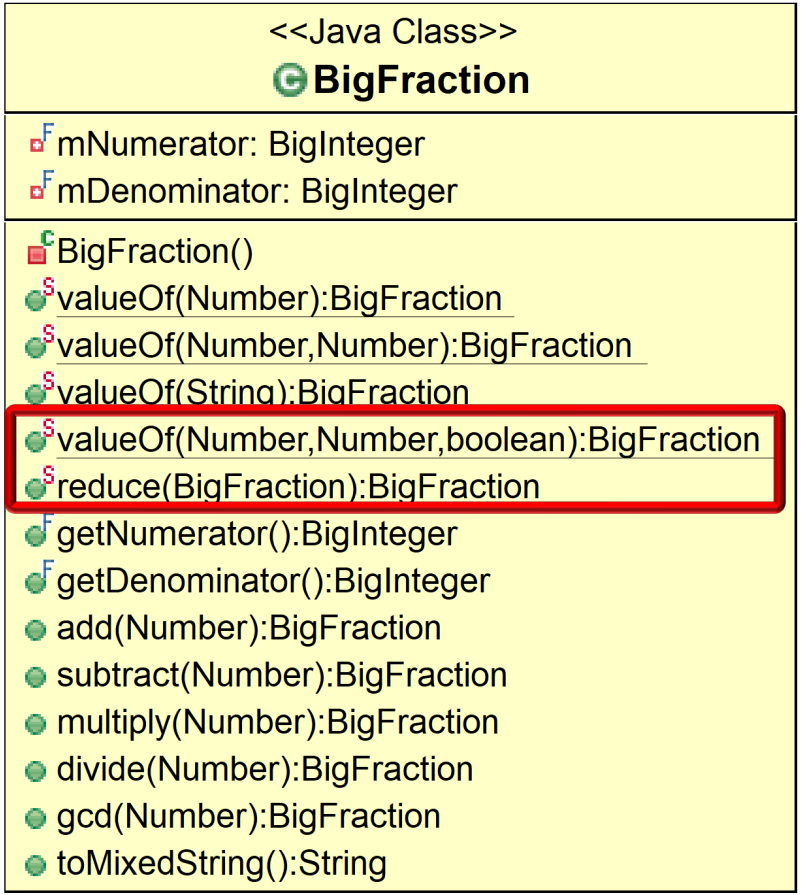
Overview of the BigFraction Class

- 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 → 4/5
    - 12/24 → 1/2
    - 144/216 → 2/3



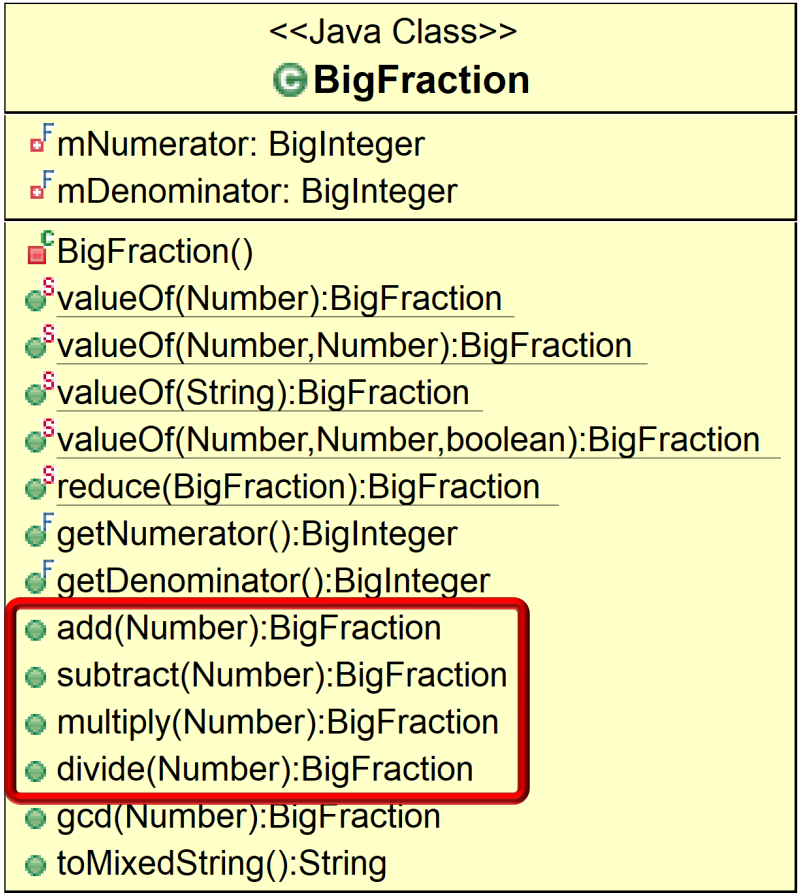
Overview of the BigFraction Class

- 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 (→ 1/2)



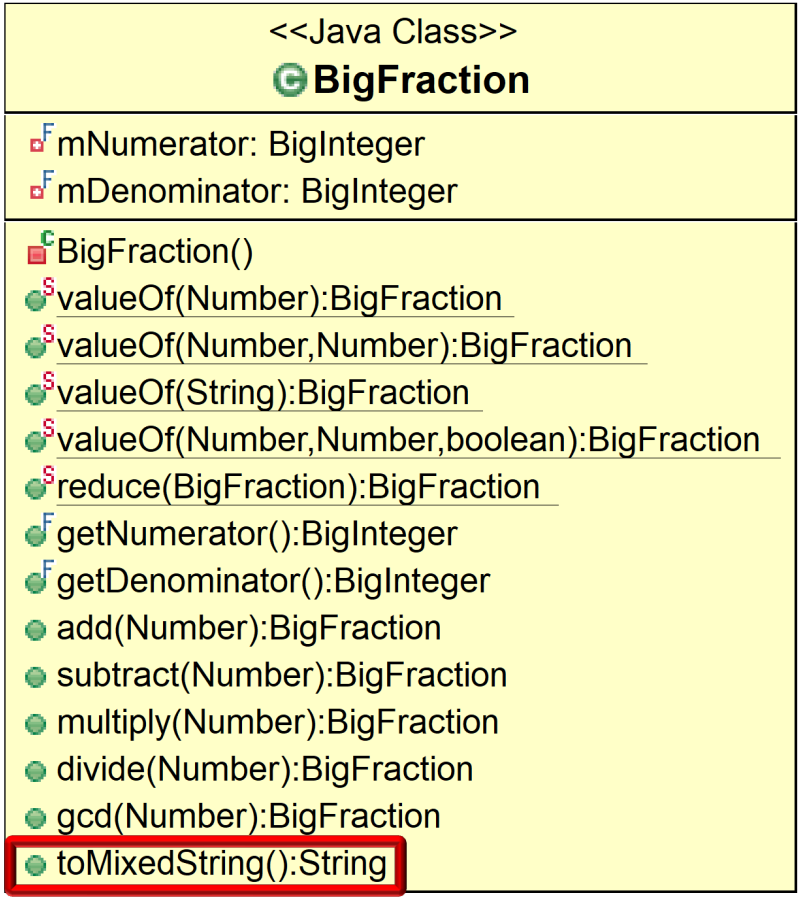
Overview of the BigFraction Class

- 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 ×2/3 = 3



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 → 4 1/2

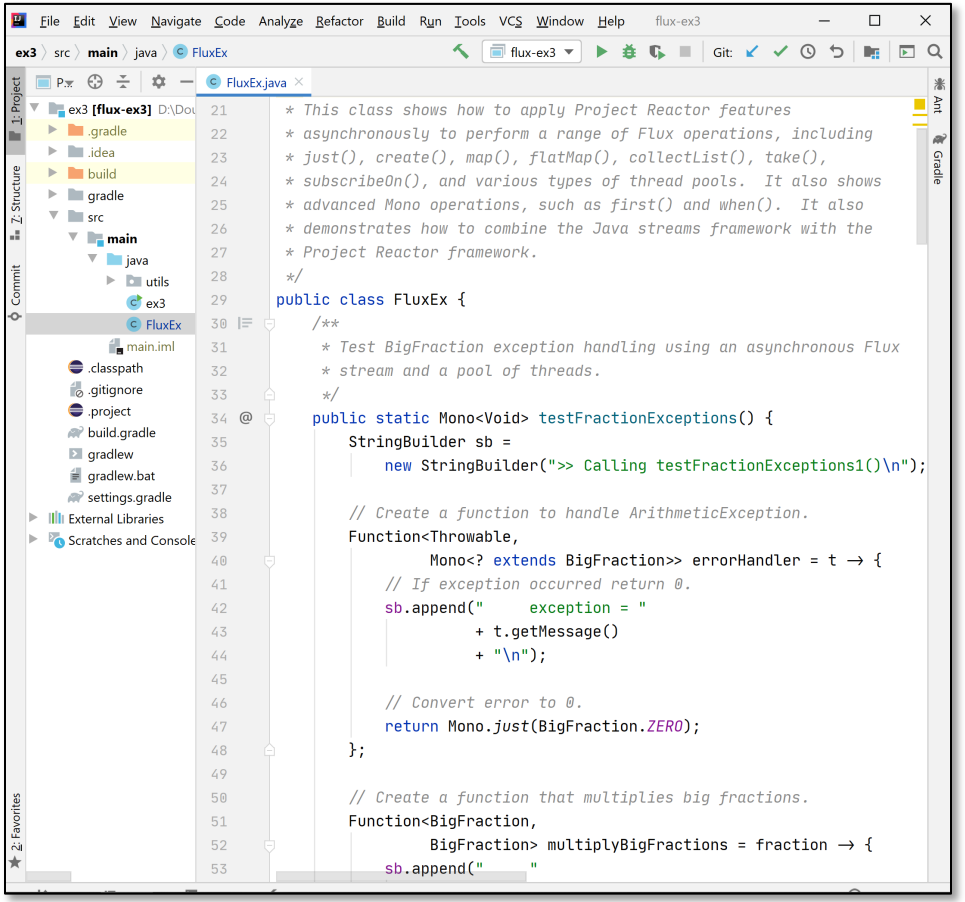


See www.mathsisfun.com/improper-fractions.html

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



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/douglascraigschmidt/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);
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Observable

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/douglascraigschmidt/LiveLessons/tree/master/Reactive/Flowable

End of Overview of the BigFraction Case Studies