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**

```
public class BigFraction {
    private BigInteger mNumerator;
    private BigInteger mDenominator;

    public BigFraction() {
        // Constructor
    }

    public static BigFraction valueOf(Number number) {
        // Value from Number
    }

    public static BigFraction valueOf(Number number, Number number2) {
        // Value from two Numbers
    }

    public static BigFraction valueOf(String string) {
        // Value from String
    }

    public static BigFraction valueOf(Number number, Number number2, boolean booleanValue) {
        // Value from Number, Number, boolean
    }

    public BigFraction reduce(BigFraction bigFraction) {
        // Reduce
    }

    public BigInteger getNumerator() {
        // Get numerator
    }

    public BigInteger getDenominator() {
        // Get denominator
    }

    public BigFraction add(Number number) {
        // Add
    }

    public BigFraction subtract(Number number) {
        // Subtract
    }

    public BigFraction multiply(Number number) {
        // Multiply
    }

    public BigFraction divide(Number number) {
        // Divide
    }

    public BigFraction gcd(Number number) {
        // Greatest Common Divisor
    }

    public String toMixedString() {
        // To mixed 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

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)

<<Java Class>>

```java
public class BigFraction {
    // Fields
    private BigInteger mNumerator;
    private BigInteger mDenominator;

    // Constructors
    public BigFraction() {
    }
    public BigFraction(BigInteger numerator, BigInteger denominator) {
    }
    public BigFraction(Number value) {
    }
    public BigFraction(Number numerator, Number denominator) {
    }
    public BigFraction(String value) {
    }

    // Factory methods
    public static BigFraction fromNumber(Number number) {
    }
    public static BigFraction fromNumber(Number numerator, Number denominator) {
    }
    public static BigFraction fromString(String value) {
    }

    // Methods
    public BigInteger getNumerator() {
    }
    public BigInteger getDenominator() {
    }
    public BigFraction add(Number other) {
    }
    public BigFraction subtract(Number other) {
    }
    public BigFraction multiply(Number other) {
    }
    public BigFraction divide(Number other) {
    }
    public int gcd(Number other) {
    }
    public String toMixedString() {
    }
    public BigFraction reduce() {
    }
    public BigFraction reduce(boolean reduce) {
    }
}
```
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
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](http://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

```java
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(bigFractionList)
    .subscribeOn(scheduler)

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

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

See [github.com/douglascairnschmidt/LiveLessons/tree/master/Reactive/Observable](https://github.com/douglascairnschmidt/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.

  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](github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Flowable)
End of Overview of the BigFraction Case Studies