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 Project Reactor API
- Understand key classes in the RxJava API
- Be aware of the structure & functionality of the BigFraction case studies

These case studies demonstrate *many* Project Reactor & RxJava features.
Overview of the BigFraction Class
Overview of the BigFraction Class

- Upcoming lessons show how to apply Project Reactor & 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 Project Reactor & 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 Project Reactor & 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 Project Reactor & 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 Project Reactor & 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 Project Reactor & 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 Project Reactor & RxJava framework features
Overview of the BigFraction Case Studies

- The Project Reactor Mono case studies show how to create, reduce, multiply, & display BigFraction objects using many Mono features
  - e.g., fromCallable(), just(), zip(), zipWith(), doOnSuccess(), first(), when(), then(), subscribeOn(), & various thread pools

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

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/mono](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/mono)
Overview of the BigFraction Case Studies

- The Project Reactor Flux case studies show how to create, reduce, multiply, & display BigFraction objects using many Flux features
  - e.g., fromIterable(), just(), map(), create(), doOnNext(), flatMap(), take(), interval(), subscribeOn(), collectList(), subscribe(), & various thread pools

```
Flux
  .create
    (bigFractionEmitter)
  .take(sMAX_FRACTIONS)
  .flatMap(unreducedFraction ->
    reduceAndMultiplyFraction
      (unreducedFraction,
       Schedulers.parallel()))
  .collectList()
  .flatMap(list ->
    BigFractionUtils
      .sortAndPrintList
        (list, sb))
```

Overview of the BigFraction Case Studies

- The Project Reactor Flux case studies show how to create, reduce, multiply, & display BigFraction objects using many Flux features
  - e.g., fromIterable(), just(), map(), create(), doOnNext(), flatMap(), take(), interval(), subscribeOn(), collectList(), subscribe(), & various thread pools
- They also demonstrate how the Java streams framework can be used together with the Project Reactor framework
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 BigFraction objects using many Observable features
  - e.g., just(), map(), create(), interval(), filter(), doOnNext(), blockingSubscribe(), take(), doOnComplete(), subscribe(), flatMap(), fromIterable(), subscribeOn(), observeOn(), range(), count(), collectList(), & various thread pools

```
return Observable
    .fromCallable(() -> BigFraction
        .reduce(unreducedFraction))
    .subscribeOn(scheduler)
    .flatMap(reducedFraction ->
        Observable
            .fromCallable(() ->
                reducedFraction.multiply(sBigReducedFraction))
            .subscribeOn(scheduler));
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable
End of Overview of the BigFraction Case Studies