Overview of the BigFraction Case Studies

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

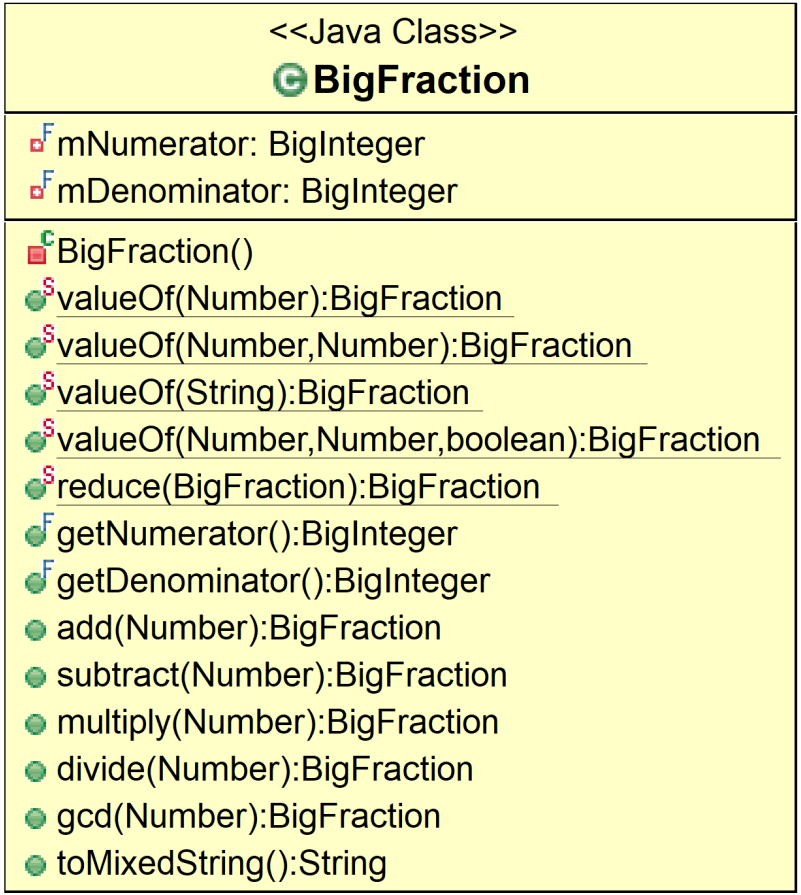
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

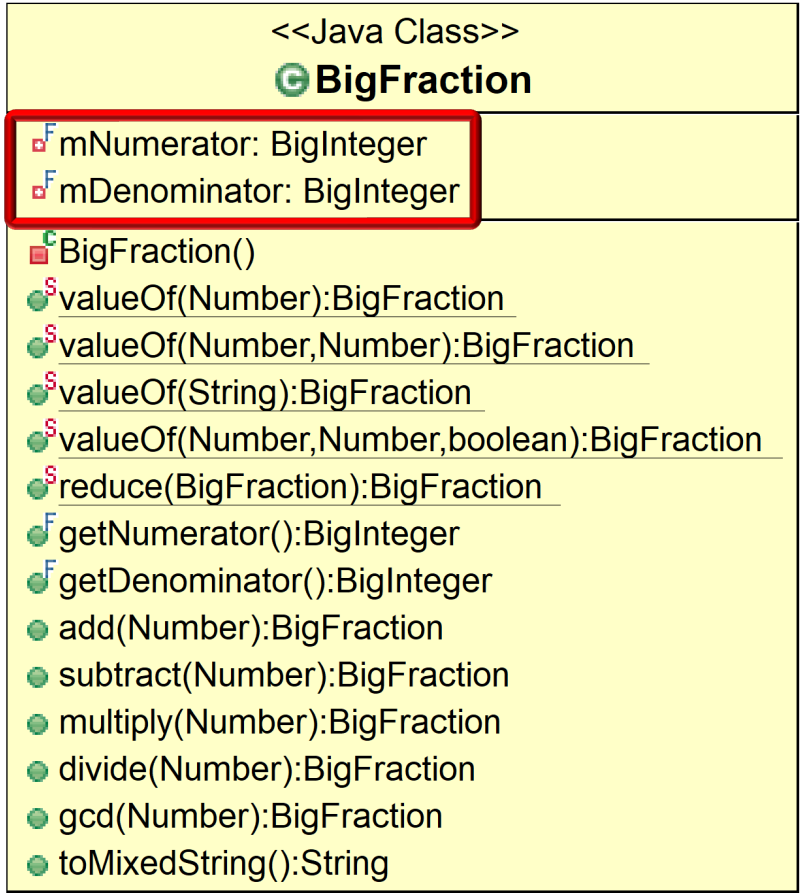
- Understand key classes in the Project Reactor API
- Be aware of the structure & functionality of the BigFraction case studies
  - These case studies showcase many operators in the Project Reactor Mono, Flux, & ParallelFlux classes



Overview of the BigFraction Class

Overview of the BigFraction Class

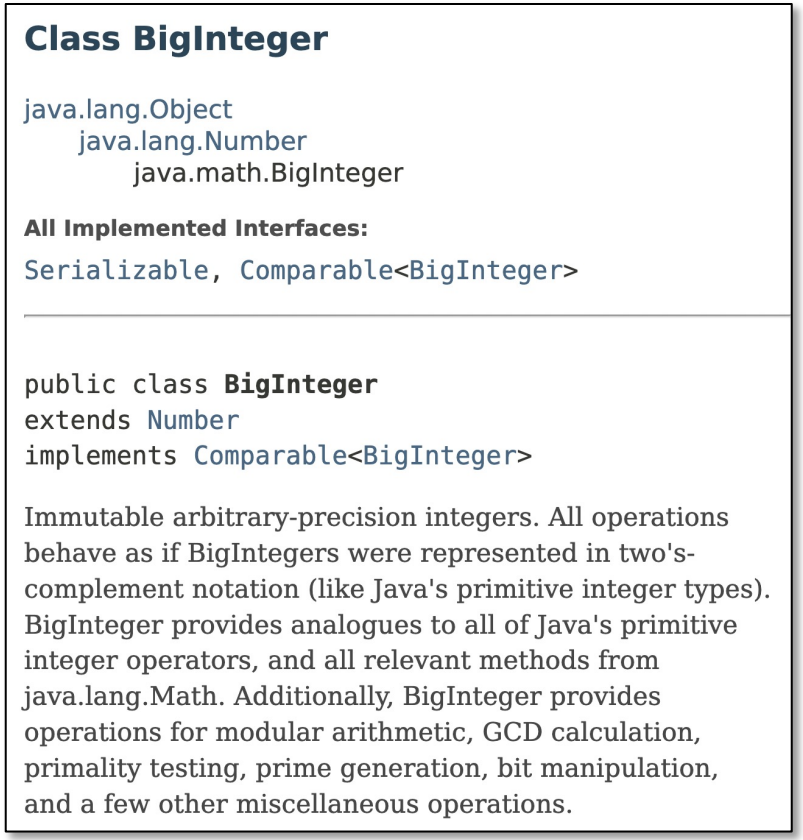
- Upcoming lessons show how to apply Project Reactor features in the context of a BigFraction class
  - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator



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

Overview of the BigFraction Class

- Upcoming lessons show how to apply Project Reactor features in the context of a BigFraction class
  - Arbitrary-precision fraction, utilizing BigIntegers for numerator & denominator
    - BigInteger provides arbitrary-precision integers & associated operators



Class BigInteger

java.lang.Object
  java.lang.Number
    java.math.BigInteger

All Implemented Interfaces:

Serializable, Comparable<BigInteger>

---

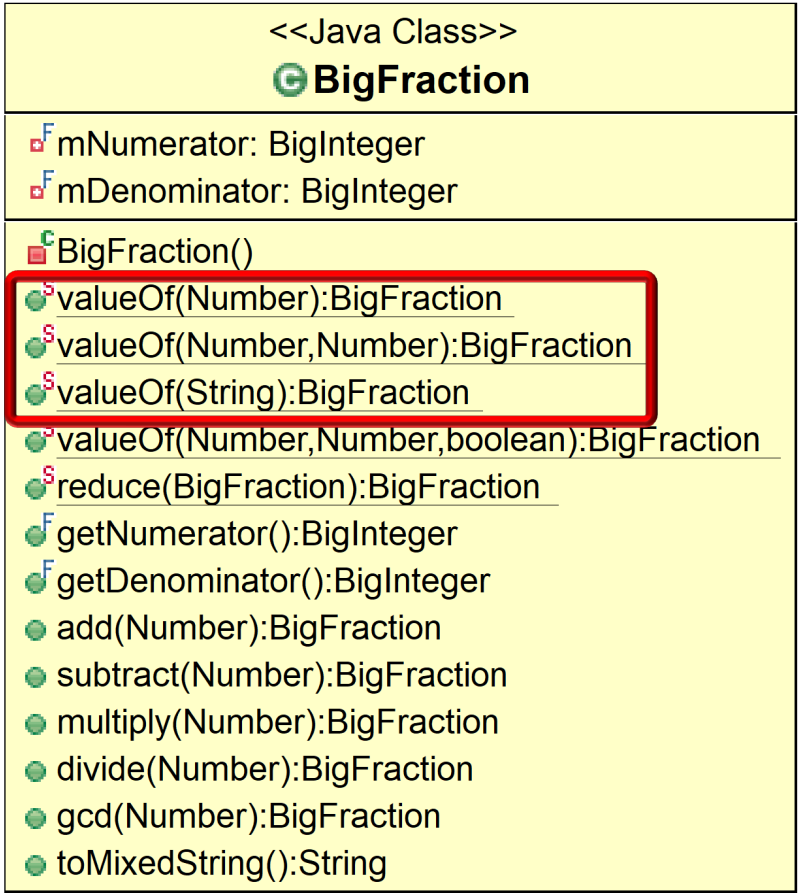
public class **BigInteger**
extends Number
implements Comparable<BigInteger>

Immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

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

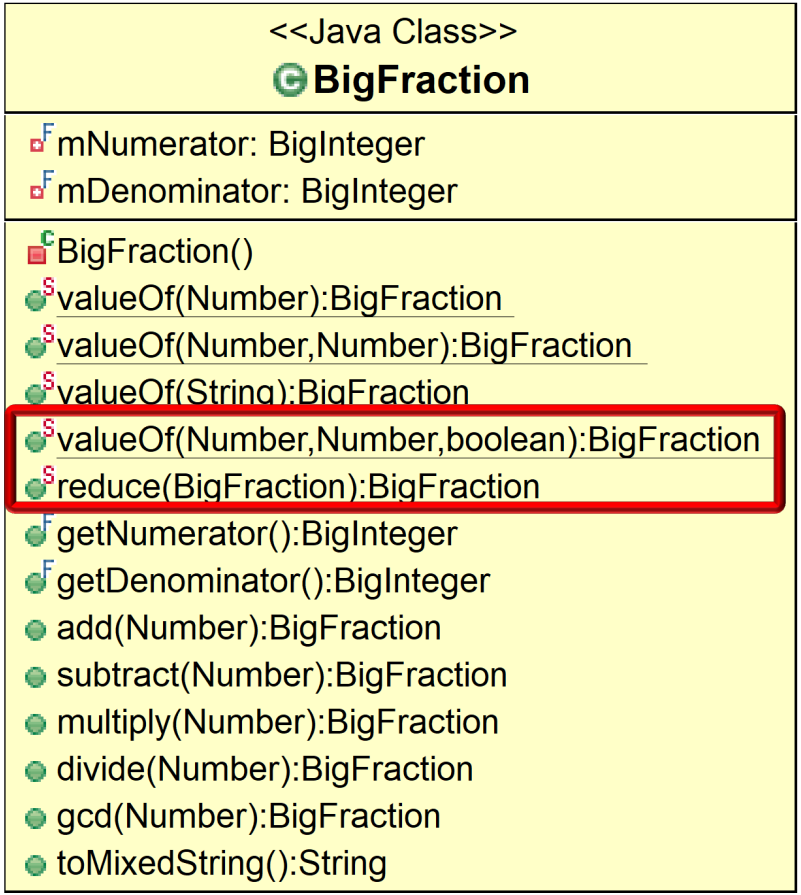
Overview of the BigFraction Class

- Upcoming lessons show how to apply Project Reactor 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



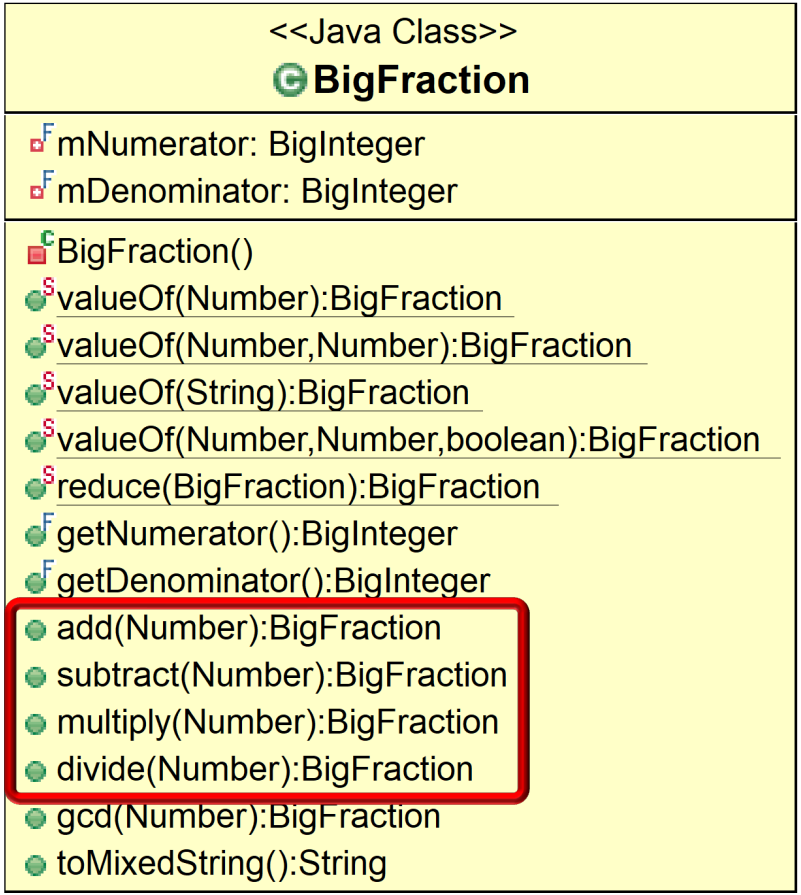
Overview of the BigFraction Class

- Upcoming lessons show how to apply Project Reactor 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)



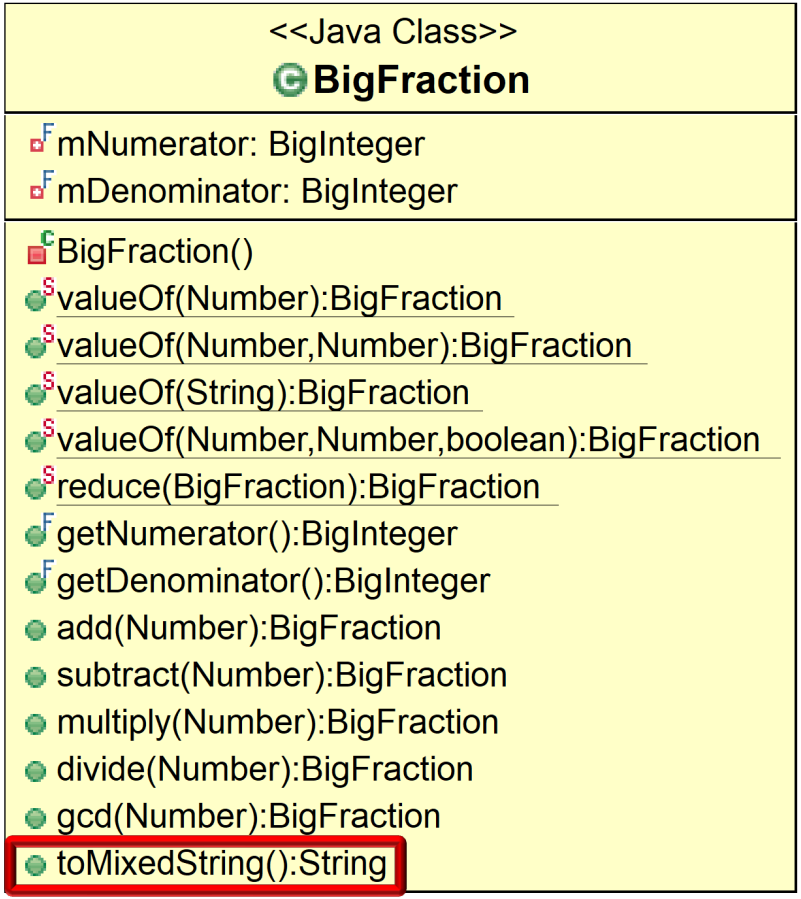
Overview of the BigFraction Class

- Upcoming lessons show how to apply Project Reactor 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 Project Reactor 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

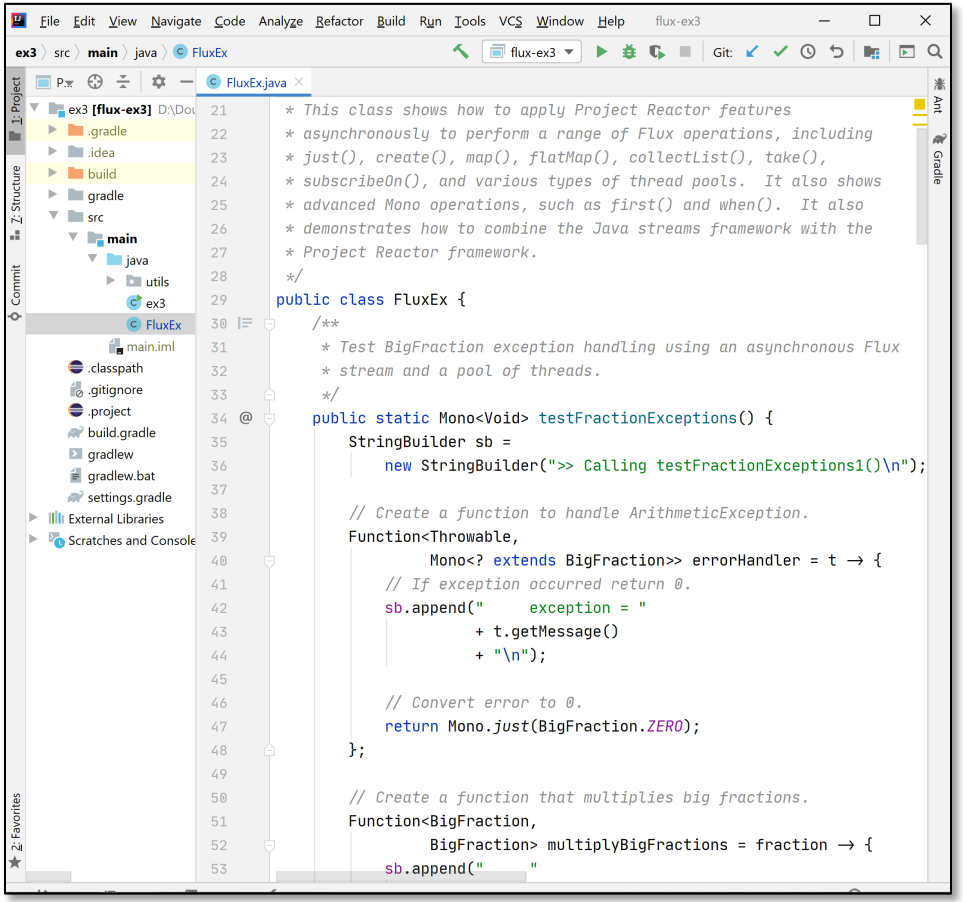


See www.mathsisfun.com/improper-fractions.html

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 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

```
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

Overview of the BigFraction Case Studies

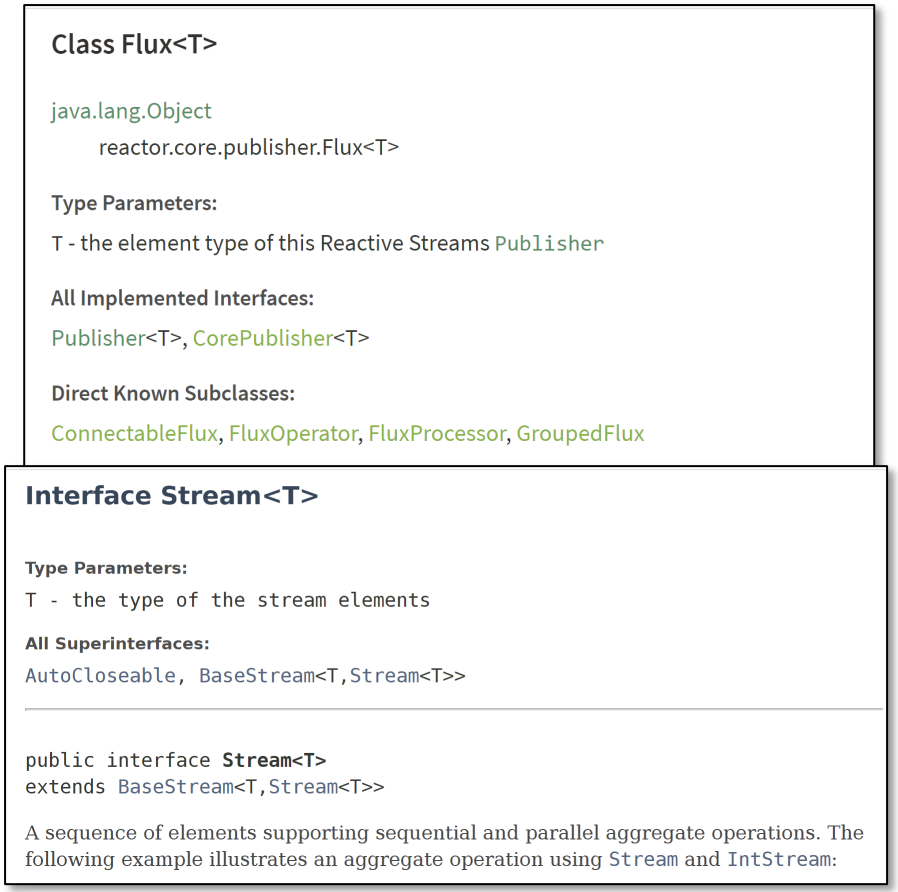
- The Project Reactor Flux case studies show how to create, reduce, multiply, & display BigFraction objects using many Flux features
  - e.g., fromIterable(), just(), map(), create(), doOnNext(), flatMap(), take(), interval(), subscribeOn(), collectList(), subscribe(), & various thread pools

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

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

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



Class Flux<T>

java.lang.Object
reactor.core.publisher.Flux<T>

Type Parameters:

T - the element type of this Reactive Streams Publisher

All Implemented Interfaces:

Publisher<T>, CorePublisher<T>

Direct Known Subclasses:

ConnectableFlux, FluxOperator, FluxProcessor, GroupedFlux

Interface Stream<T>

Type Parameters:

T - the type of the stream elements

All Superinterfaces:

AutoCloseable, BaseStream<T,Stream<T>>

public interface Stream<T>
extends BaseStream<T,Stream<T>>

A sequence of elements supporting sequential and parallel aggregate operations. The following example illustrates an aggregate operation using Stream and IntStream:

End of Overview of the BigFraction Case Studies