Applying Key Operators in the Observable Class: Case Study ex2 (Part 1)

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

• Part 1 of this case study explores how to use the RxJava Observable operators map(), create(), interval(), ignoreElements(), doOnNext(), take(), subscribeOn(), subscribe(), filter(), & doOnComplete() to create large random BigInteger objects & check asynchronously if they are prime in a background thread.

```java
Observable
    .create
    (ObservableEx::emitInterval)
    .map(bigInteger ->
        ObservableEx.checkIfPrime
            (bigInteger, sb))
...
    .doOnComplete(() ->
        BigFractionUtils.
        display(sb.toString()))
    .ignoreElements();
```

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex2](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex2)
Applying Key Operators in the Observable Class to ex2
Applying Key Operators in the Flux Class to ex2

```java
public static Mono<Void> testIsPrimeTimed() {
    StringBuffer sb =
        new StringBuffer(">> Calling testIsPrimeTimed()\n");

    return Flux
        // Factory method creates a flow of random big integers
        // that are generated at a periodic interval in a
        // background thread.
        .create(makeTimedFluxSink(sb))
        .Flux<BigInteger>

        // Use a memoizer to check if each random big integer is
        // prime or not on the background thread.
        .map(bigInteger ->
            FluxEx.checkIfPrime(bigInteger, sb))
        .Flux<FluxEx.PrimeResult>

        // Process each big integer on the background thread.
        .doOnNext(bigInteger ->
            FluxEx.processResult(bigInteger,
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex2](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex2)
End of Applying Key Methods in the Observable Class: Case Study ex2 (Part 1)