Applying Key Operators in the Flux Class: Case Study ex1 (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 case study ex1 shows how to use Flux operators fromIterable(), just(), fromArray(), from(), map(), doOnNext(), mergeWith(), repeat(), & subscribe() to create, reduce, multiply, & display BigFraction objects synchronously.

```java
Flux
.just(BigFraction.valueOf(100,3),
    BigFraction.valueOf(100,4),
    BigFraction.valueOf(100,2),
    BigFraction.valueOf(100,1))
.map(fraction -> fraction
    .multiply(sBigReducedFraction))
.subscribe
    (fraction -> sb.append(" = "+fraction.toString() + "\n"),
    error -> sb.append("error"),
    () -> BigFractionUtils.display(sb.toString()));
```
Applying Key Operators in the Flux Class to ex1
Applying Key Operators in the Flux Class to ex1

```java
public class FluxEx {
    /**
     * This class shows how to apply Project Reactor features
     * synchronously to perform basic Flux operations, including just(),
     * map(), and subscribe().
     * https://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html
     */

    public static Mono<Void> testFractionMultiplication() {
        StringBuilder sb =
            new StringBuilder("\n\n>> Calling testFractionMultiplication()\n\nFlux

    // Use just() to generate a stream of big fractions.
    // https://projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux
    .just(BigFraction.valueOf(numerator: 100, denominator: 3),
        BigFraction.valueOf(numerator: 100, denominator: 4),
        BigFraction.valueOf(numerator: 100, denominator: 2),
        BigFraction.valueOf(numerator: 100, denominator: 1))
    .map(b -> b.multiply(b))
    .subscribe(System.out::println);
    return Flux.just();
    }
}
```

End of Applying Key Operators in the Flux Class: Case Study ex1 (Part 1)