Applying Key Operators in Project Reactor: Case Study ex6

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

- Case study ex6 shows how to integrate Java Streams operations generate(), limit(), map(), & collect() with Project Reactor Mono operators fromCallable(), then(), materialize(), firstWithSignal(), map(), flatMap(), flatMapIterable(), subscribeOn(), & when() to create, reduce, multiply, & display BigFraction objects asynchronously

```
return Stream
    .generate(() ->
        makeBigFraction(sRANDOM,
                        false))

    .limit(sMAX_FRACTIONS)

    .map(unreducedBigFraction ->
        reduceAndMultiplyFraction
        (unreducedBigFraction,
         Schedulers.fromExecutor
         (ForkJoinPool
          .commonPool())));
```
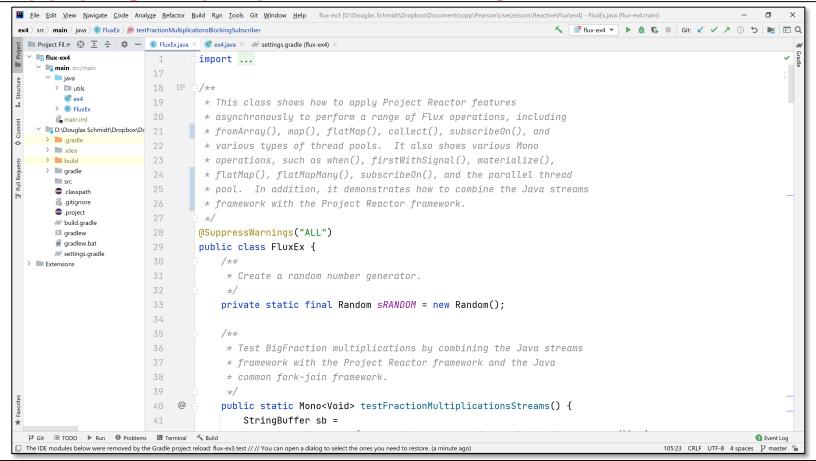
Learning Objectives in this Part of the Lesson

- Case study ex6 shows how to integrate Java Streams operations generate(), limit(), map(), & collect() with Project Reactor Mono operators fromCallable(), then(), materialize(), firstWithSignal(), map(), flatMap(), flatMapIterable(), subscribeOn(), & when() to create, reduce, multiply, & display BigFraction objects asynchronously
  - It also shows how to implement two Java Streams Collectors for asynchronous Mono objects

```
return monos -> Mono
       .when(monos)

       .materialize()

       .flatMap(v -> Flux
              .fromIterable(monos)

              .map(Mono::block)

       .collect(toList())));
```

Applying Key Operators in Project Reactor to ex6

Applying Key Operators in Project Reactor to ex6

End of Applying Key Methods in Project Reactor: Case Study ex6