Applying Key Operators in Project Reactor: Case Study ex4 (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 ex4 applies Flux operators `subscribe()`, `flatMap()`, & `fromArray()` to create, multiply, & display BigFraction objects asynchronously.

```java
Mono
  .fromCallable(() ->
    .makeBigFraction
      (sRANDOM, true))
  .flatMapMany(bf1 -> Flux
    .fromArray(bigFractionArray)
    .flatMap(bf2 -> Mono
      .fromCallable(() -> bf2)
      .subscribeOn(scheduler)
      .map(___ -> bf2
        .multiply(bf1))))

.subscribe(blockingSubscriber);
```

Learning Objectives in this Part of the Lesson

- Part 1 of case study ex4 applies Flux operators `subscribe()`, `flatMap()`, & `fromArray()` to create, multiply, & display BigFraction objects asynchronously.

- It also shows how to use Mono operators `fromCallable()`, `map()`, `flatMapMany()`, & `subscribeOn()`.
Learning Objectives in this Part of the Lesson

• Part 1 of case study ex4 applies Flux operators subscribe(), flatMap(), & fromArray() to create, multiply, & display BigFraction objects asynchronously

• It also shows how to use Mono operators fromCallable(), map(), flatMapMany(), & subscribeOn()

• In addition, it shows how to create & use a generic blocking Subscriber

• Can be applied to workaround the lack of a blockingSubscribe() operator

class BlockingSubscriber<T> {
    implements Subscriber<T> {
        final CountDownLatch mLatch;
        ...
        @Override
        public void onComplete() {
            ...
            mLatch.countDown();
        }
        ...
    }
}
Applying Key Operators in Project Reactor to ex4
Applying Key Operators in Project Reactor to ex4

```java
import ...

/**
 * This class shows how to apply Project Reactor features
 * asynchronously to perform a range of Flux operations, including
 * `fromArray()`, `map()`, `flatMap()`, `collect()`, `subscribeOn()`, and
 * various types of thread pools. It also shows various Mono
 * operations, such as `when()`, `firstWithSignal()`, `materialize()`,
 * `flatMap()`, `flatMapMany()`, `subscribeOn()`, and the parallel thread
 * pool. In addition, it demonstrates how to combine the Java streams
 * framework with the Project Reactor framework.
 */

@SuppressWarnings("ALL")
public class FluxEx {
    /**
     * Create a random number generator.
     */
    private static final Random sRANDOM = new Random();

    /**
     * Test BigFraction multiplications by combining the Java streams
     * framework with the Project Reactor framework and the Java
     * common fork-join framework.
     */
    public static Mono<Void> testFractionMultiplicationsStreams() {
        StringBuffer sb =
```
End of Applying Key Methods in Project Reactor: Case Study ex4 (Part 1)