

Applying Key Operators in the Flux Class: Case Study ex2 (Part 2)

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 2 of case study ex2 shows how to use Flux operators `create()`, `map()`, `filter()`, `take()`, `subscribe()`, `subscribeOn()`, `publishOn()`, `then()`, `range()`, `doOnNext()`, & `doFinally()` to create large random `BigInteger` objects & asynchronously check if they are prime via publisher & subscriber threads created using `Schedulers.newParallel()`

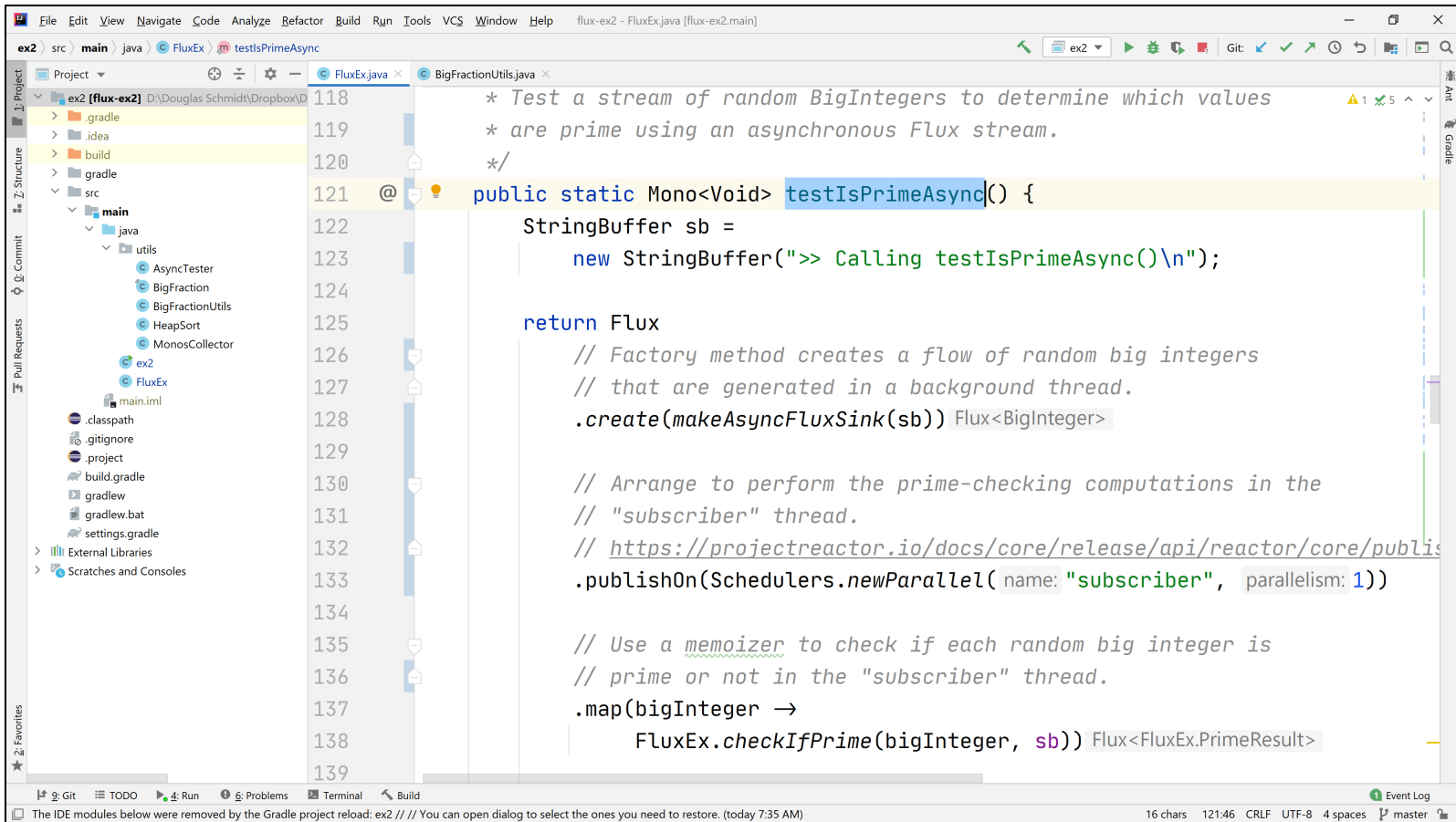
```
Scheduler publisher = Schedulers
    .newParallel("publisher", 1);
```

```
Flux
```

```
    .range(1, sMAX_ITERATIONS)
    ...
    .subscribeOn(publisher)
    .map(__ -> BigInteger
        .valueOf(lowerBound + rand
            .nextInt(sMAX_ITERATIONS)))
    ...
    .doFinally(() -> publisher
        .dispose())
    .subscribe(sink::next,
        err -> ...,
        sink::complete);
```

Applying Key Operators in the Flux Class to ex2

Applying Key Operators in the Flux Class to ex2



```
118      * Test a stream of random BigIntegers to determine which values
119      * are prime using an asynchronous Flux stream.
120      */
121      @Test public static Mono<Void> testIsPrimeAsync() {
122          StringBuffer sb =
123              new StringBuffer(">> Calling testIsPrimeAsync()\n");
124
125          return Flux
126              // Factory method creates a flow of random big integers
127              // that are generated in a background thread.
128              .create(makeAsyncFluxSink(sb)) Flux<BigInteger>
129
130              // Arrange to perform the prime-checking computations in the
131              // "subscriber" thread.
132              // https://projectreactor.io/docs/core/release/api/reactor/core/publish
133              .publishOn(Schedulers.newParallel("subscriber", parallelism: 1))
134
135              // Use a memoizer to check if each random big integer is
136              // prime or not in the "subscriber" thread.
137              .map(bigInteger ->
138                  FluxEx.checkIfPrime(bigInteger, sb)) Flux<FluxEx.PrimeResult>
139
```

See github.com/douglas-craig-schmidt/LiveLessons/tree/master/Reactive/flux/ex2

End of Applying Key Methods in the Flux Class: Case Study ex2 (Part 2)