

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

Observable

```
.create
    (ObservableEx::emitInterval)

.map (bigInteger ->
    ObservableEx.checkIfPrime
        (bigInteger, sb))

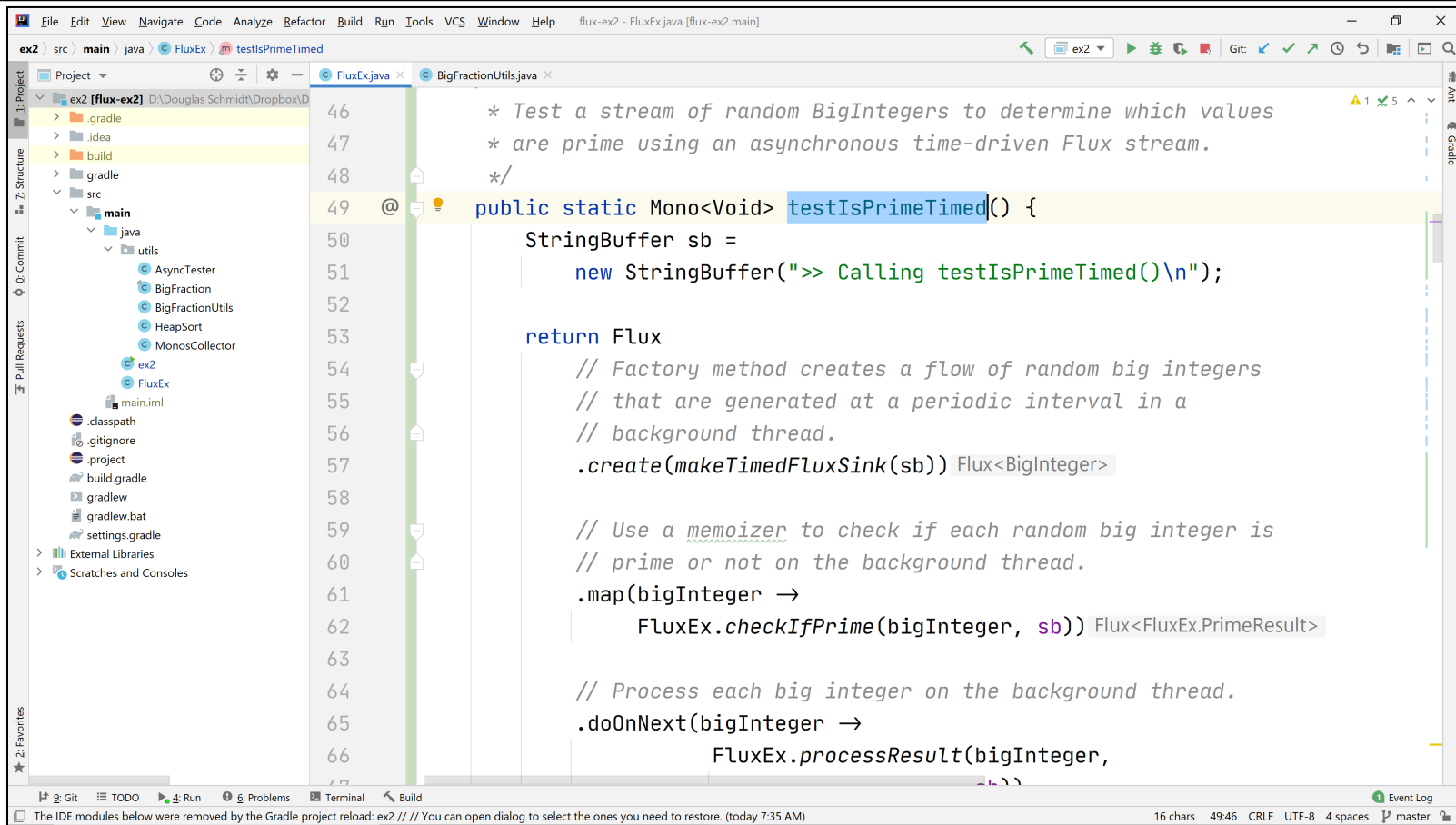
...

.doOnComplete ( () ->
    BigFractionUtils.
        display (sb.toString()) )

.ignoreElements () ;
```

Applying Key Operators in the Observable Class to ex2

Applying Key Operators in the Flux Class to ex2



The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Displays the project structure for 'ex2 [flux-ex2]'. The 'main' directory contains a 'java' subdirectory with 'utils'. Under 'utils', there are classes: 'AsyncTester', 'BigFraction', 'BigFractionUtils', 'HeapSort', and 'MonosCollector'. The 'ex2' directory contains 'FluxEx' and 'main.iml'. Other files include '.classpath', '.gitignore', 'project', 'build.gradle', 'gradlew', 'gradlew.bat', and 'settings.gradle'.
- Code Editor (Center):** Shows the implementation of the `testIsPrimeTimed()` method in `FluxEx.java`. The code is as follows:

```
46      * Test a stream of random BigIntegers to determine which values
47      * are prime using an asynchronous time-driven Flux stream.
48      */
49      @Test
50      public static Mono<Void> testIsPrimeTimed() {
51          StringBuffer sb =
52              new StringBuffer(">> Calling testIsPrimeTimed()\n");
53
54          return Flux
55              // Factory method creates a flow of random big integers
56              // that are generated at a periodic interval in a
57              // background thread.
58              .create(makeTimedFluxSink(sb)) Flux<BigInteger>
59
60              // Use a memoizer to check if each random big integer is
61              // prime or not on the background thread.
62              .map(bigInteger →
63                  FluxEx.checkIfPrime(bigInteger, sb)) Flux<FluxEx.PrimeResult>
64
65              // Process each big integer on the background thread.
66              .doOnNext(bigInteger →
67                  FluxEx.processResult(bigInteger,
```
- Bottom Bar:** Shows the status of the IDE, including 'Git', 'TODO', 'Run', 'Problems', 'Terminal', and 'Build'. A message at the bottom states: 'The IDE modules below were removed by the Gradle project reload: ex2 // You can open dialog to select the ones you need to restore. (today 7:35 AM)'. The right side of the bar shows '16 chars', '49:46', 'CRLF', 'UTF-8', '4 spaces', and 'master'.

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex2

End of Applying Key Methods in the Observable Class: Case Study ex2 (Part 1)