

Applying Key Operators in the Observable

Class: Case Study ex3 (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 ex3 shows how to use RxJava Observable operators `map()`, `fromCallable()`, `generate()`, `flatMap()`, `collectInfo()`, `subscribeOn()`, `take()`, & `Schedulers.computation()` to create, reduce, multiply, & display `BigFraction` objects asynchronously using the `flatMap()` concurrency idiom

```
return Observable
    .generate(emitter)

    .take(sMAX_FRACTIONS)

    .flatMap(unreducedFraction ->
        reduceAndMultiplyFraction
            (unreducedFraction,
             Schedulers
              .computation()))

    .collect(toList())

    .flatMapCompletable(list ->
        sortAndPrintList(list, sb));
```

Learning Objectives in this Part of the Lesson

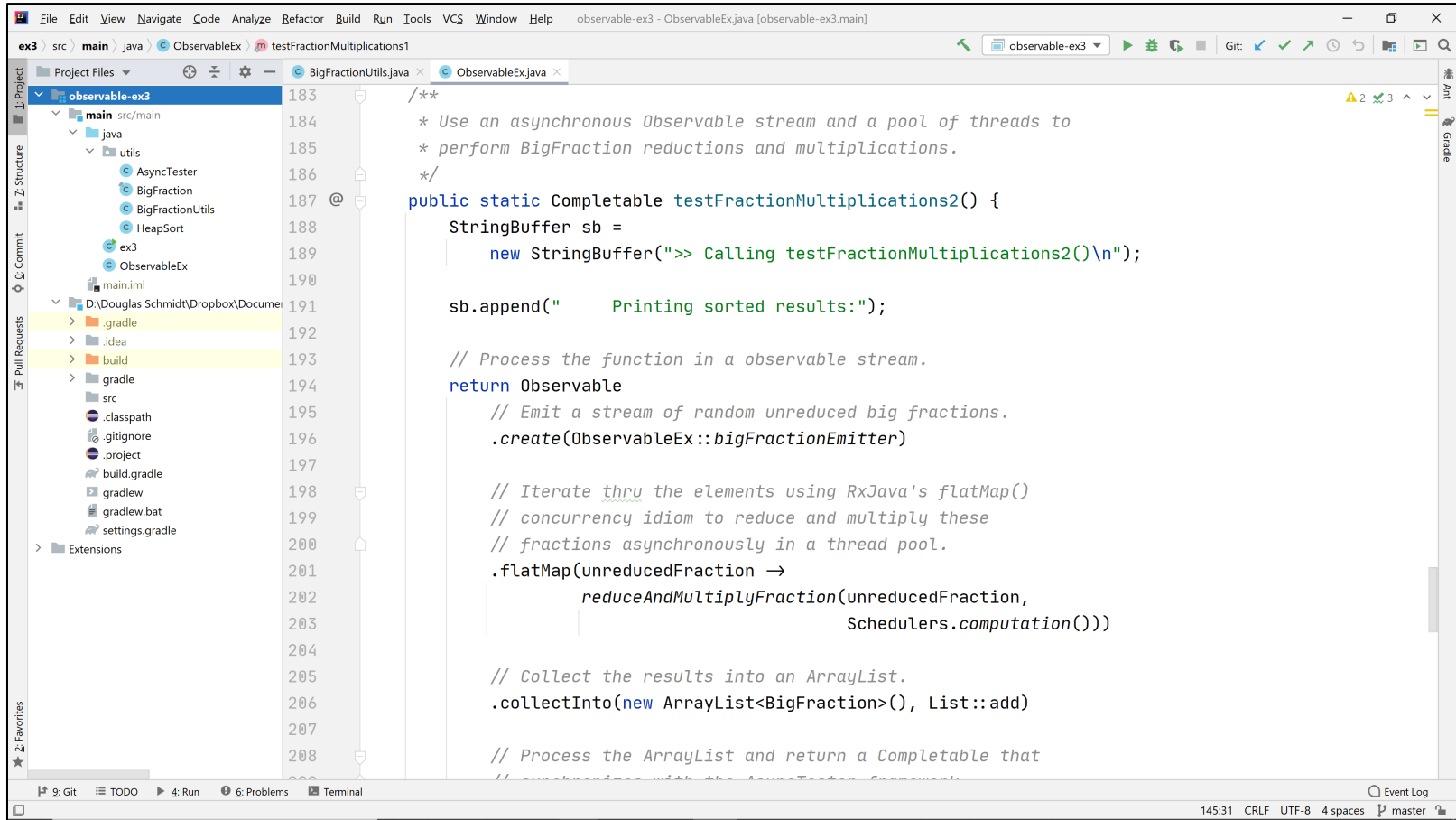
- Part 2 of case study ex3 shows how to use RxJava Observable operators `map()`, `fromCallable()`, `generate()`, `flatMap()`, `collectInfo()`, `subscribeOn()`, `take()`, & `Schedulers.computation()` to create, reduce, multiply, & display `BigFraction` objects asynchronously using the `flatMap()` concurrency idiom
- It also shows how these operators can be used with `Single` operators `fromCallable()`, `doOnSuccess()`, `flatMapCompletable()`, `ambArray()`, `ignoreElement()`, & `subscribeOn()`

```
Single<List<BigFraction>> qSort
    = Single.fromCallable(() ->
                                quickSort(list))
        .subscribeOn(Schedulers
                        .computation());
Single<List<BigFraction>> hSort
    = Single.fromCallable(() ->
                                heapSort(list))
        .subscribeOn(Schedulers
                        .computation());

return Single.ambArray
                (qSort, hSort)
        .doOnSuccess(displayList)
        .ignoreElement();
```

Applying Key Operators in the Observable Class to ex3

Applying Key Operators in the Observable Class to ex3



The screenshot shows an IDE window titled 'observable-ex3 - ObservableEx.java [observable-ex3.main]'. The left sidebar displays a project structure for 'observable-ex3' with a 'main' directory containing 'src/main/java' and 'utils'. The 'ObservableEx.java' file is open in the editor. The code defines a static method 'testFractionMultiplications2()' that uses RxJava's 'Observable' and 'Completable' classes. It creates a 'StringBuilder' to log the start of the test, then processes a stream of 'unreducedFraction' objects using 'flatMap' and 'reduceAndMultiplyFraction'. The results are collected into an 'ArrayList' and the method returns a 'Completable'.

```
183 /**
184  * Use an asynchronous Observable stream and a pool of threads to
185  * perform BigFraction reductions and multiplications.
186  */
187 @
188 public static Completable testFractionMultiplications2() {
189     StringBuffer sb =
190         new StringBuffer(">> Calling testFractionMultiplications2()\n");
191
192     sb.append("    Printing sorted results:");
193
194     // Process the function in a observable stream.
195     return Observable
196         // Emit a stream of random unreduced big fractions.
197         .create(ObservableEx::bigFractionEmitter)
198
199         // Iterate thru the elements using RxJava's flatMap()
200         // concurrency idiom to reduce and multiply these
201         // fractions asynchronously in a thread pool.
202         .flatMap(unreducedFraction ->
203             reduceAndMultiplyFraction(unreducedFraction,
204                 Schedulers.computation()))
205
206         // Collect the results into an ArrayList.
207         .collectInto(new ArrayList<BigFraction>(), List::add)
208
209         // Process the ArrayList and return a Completable that
210         // completes with the AsyncTester framework.
```

See github.com/douglasraigschmidt/LiveLessons/tree/master/Reactive/Observable/ex3

End of Applying Key Methods in the Observable Class: Case Study ex3 (Part 2)