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));
```

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

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()

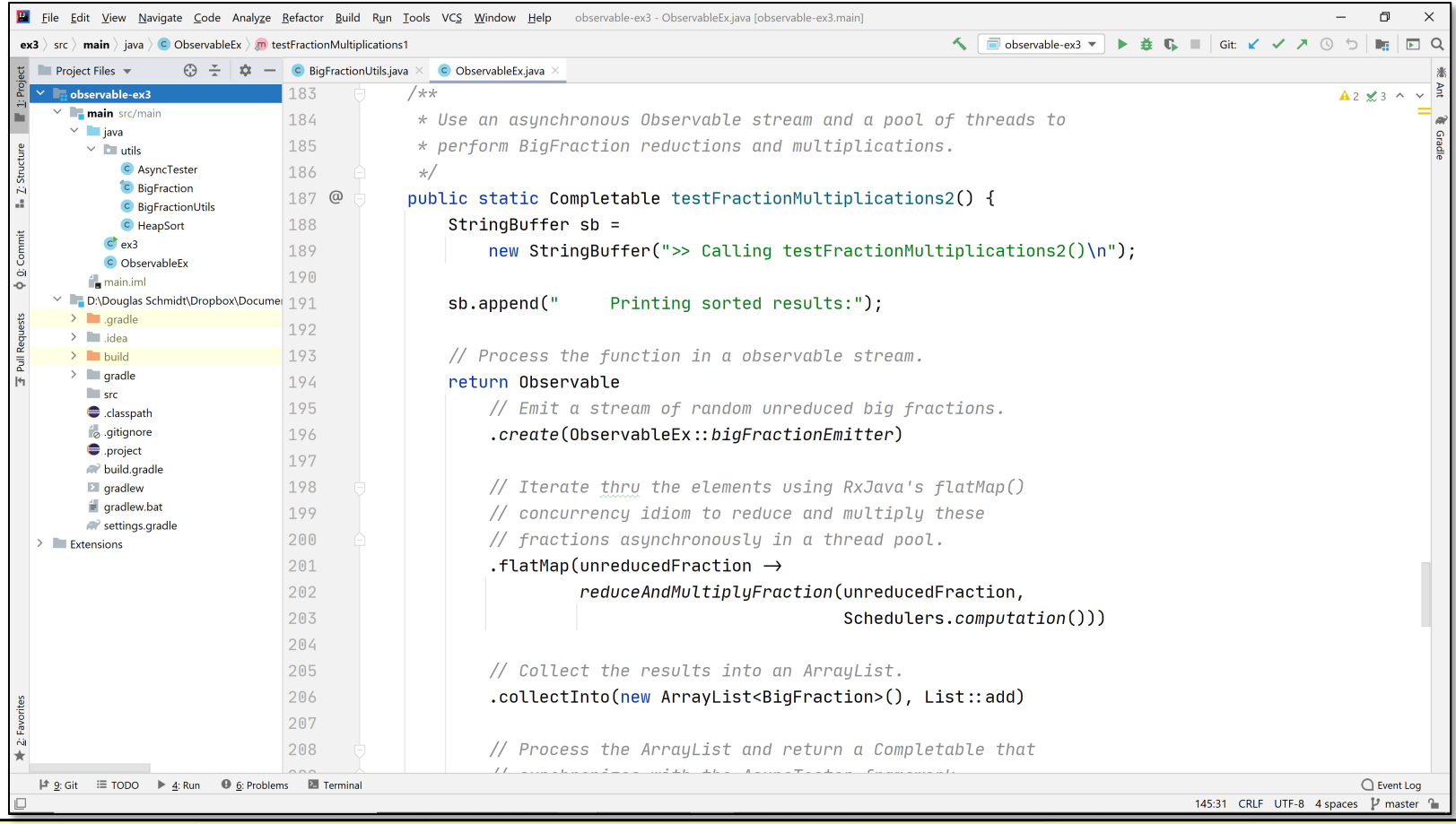
```
Single<List<BigFraction>> qSort
  = Single.fromCallable(() ->
                    quickSort(list))
    .subscribeOn(Schedulers
                    .computation());
Single<List<BigFraction>> hSort
  = Single.fromCallable(() ->
                     heapSort(list))
    .subscribeOn(Schedulers
     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

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

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

Applying Key Operators in the Observable Class to ex3

Applying Key Operators in the Observable Class to ex3



```
/**
 * Use an asynchronous Observable stream and a pool of threads to
 * perform BigFraction reductions and multiplications.
 */
public static Completable testFractionMultiplications2() {
    StringBuffer sb =
        new StringBuffer(">> Calling testFractionMultiplications2()\n");

    sb.append("     Printing sorted results:");

    // Process the function in a observable stream.
    return Observable
        // Emit a stream of random unreduced big fractions.
        .create(ObservableEx::bigFractionEmitter)

        // Iterate thru the elements using RxJava's flatMap()
        // concurrency idiom to reduce and multiply these
        // fractions asynchronously in a thread pool.
        .flatMap(unreducedFraction ->
                reduceAndMultiplyFraction(unreducedFraction,
                                          Schedulers.computation()))

        // Collect the results into an ArrayList.
        .collectInto(new ArrayList<BigFraction>(), List::add)

        // Process the ArrayList and return a Completable that
```

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

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