

Applying Key Operators in the Observable

Class: Case Study ex3 (Part 1)

Douglas C. Schmidt

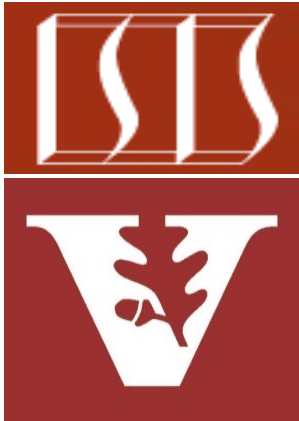
d.schmidt@vanderbilt.edu

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 ex3 shows how to use RxJava Observable operators flatMap(), fromArray(), fromIterable(), onErrorReturn(), fromCallable(), collect(), onErrorResumeNext(), reduce(), filter(), subscribeOn(), map(), & Schedulers.computation() to asynchronously create, multiply, & display BigFraction objects, even in the presence of errors

Observable

```
.fromCallable(() ->
    BigFraction
        .valueOf(Math.abs
            (sRAND.nextInt()),
            denominator))
.subscribeOn
    (Schedulers.computation())
.onErrorReturn(errorHandler)

.map(bf -> bf.multiply
    (sBigReducedFraction))
```

Learning Objectives in this Part of the Lesson

- Part 1 of case study ex3 shows how to use RxJava Observable operators flatMap(), fromArray(), fromIterable(), onErrorReturn(), fromCallable(), collect(), onErrorResumeNext(), reduce(), filter(), subscribeOn(), map(), & Schedulers .computation() to asynchronously create, multiply, & display BigFraction objects, even in the presence of errors
- It also shows how Single operators doOnSuccess(), ignoreElement(), flatMapCompletable(), & just() can be used with Observable operators

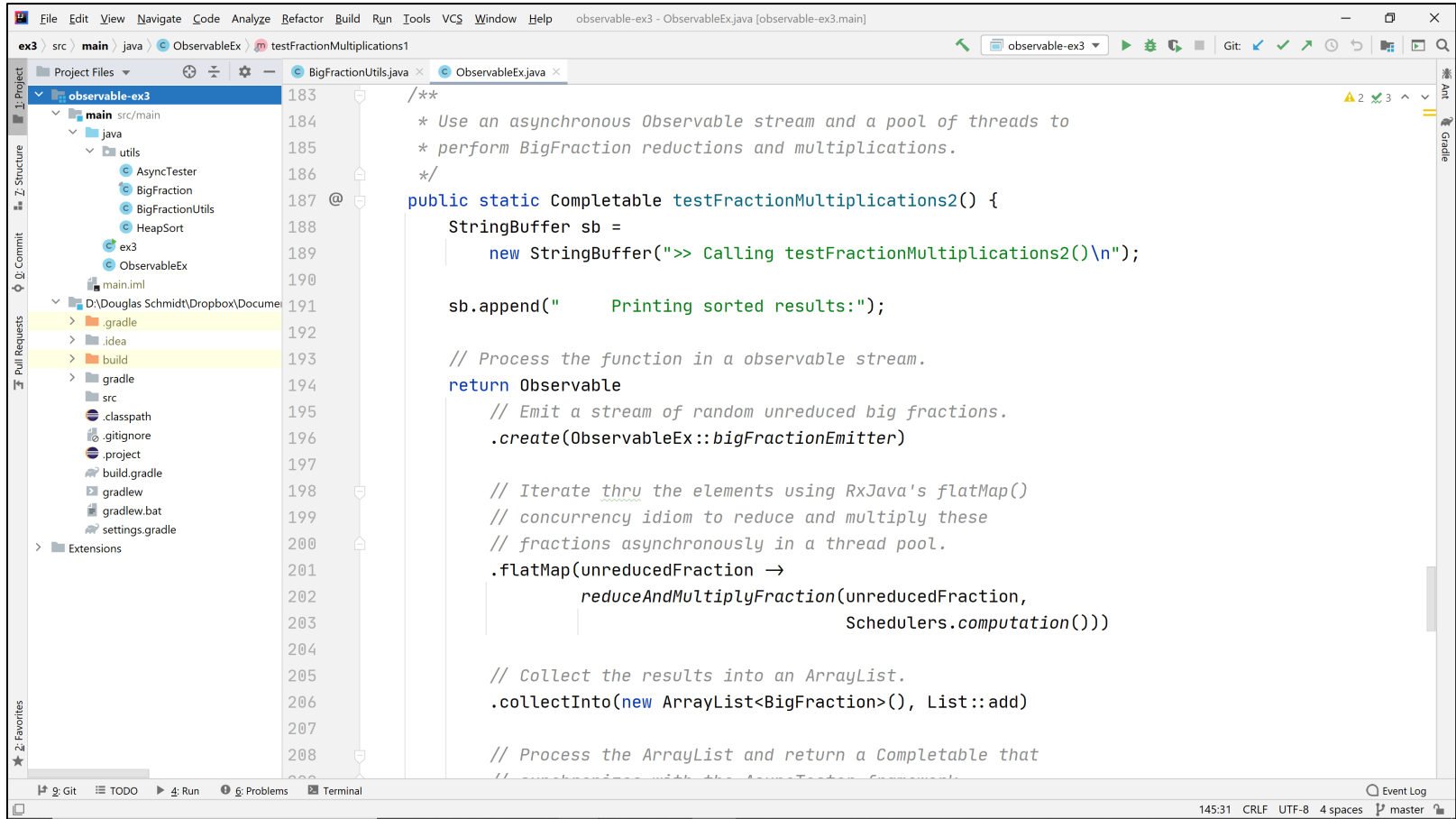
```
return Single
    .just(list)

    .doOnSuccess (displayList)

    .ignoreElement () ;
```

Applying Key Operators in the Observable Class to ex3

Applying Key Operators in the Observable Class to ex3



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

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

End of Applying Key
Operators in the Observable
Class: Case Study ex3
(Part 1)