Applying Key Operators in the Flux 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 Flux operators fromIterable(), flatMap(), map(), onErrorResume(), onErrorStop(), collectList(), filter(), onErrorContinue(), & the parallel thread pool to create, reduce, multiply, & display BigFraction objects (a)synchronously

```
return Flux
   .fromIterable(denominators)

   .map(denominator -> BigFraction
          .valueOf(Math.abs
                   (sRAND.nextInt()),
                    denominator)

   .onErrorResume(errorHandler)

   .onErrorStop()

   .collectList()

   .flatMap(list -> BigFractionUtils
             .sortAndPrintList(list,
                               sb));
```

Learning Objectives in this Part of the Lesson

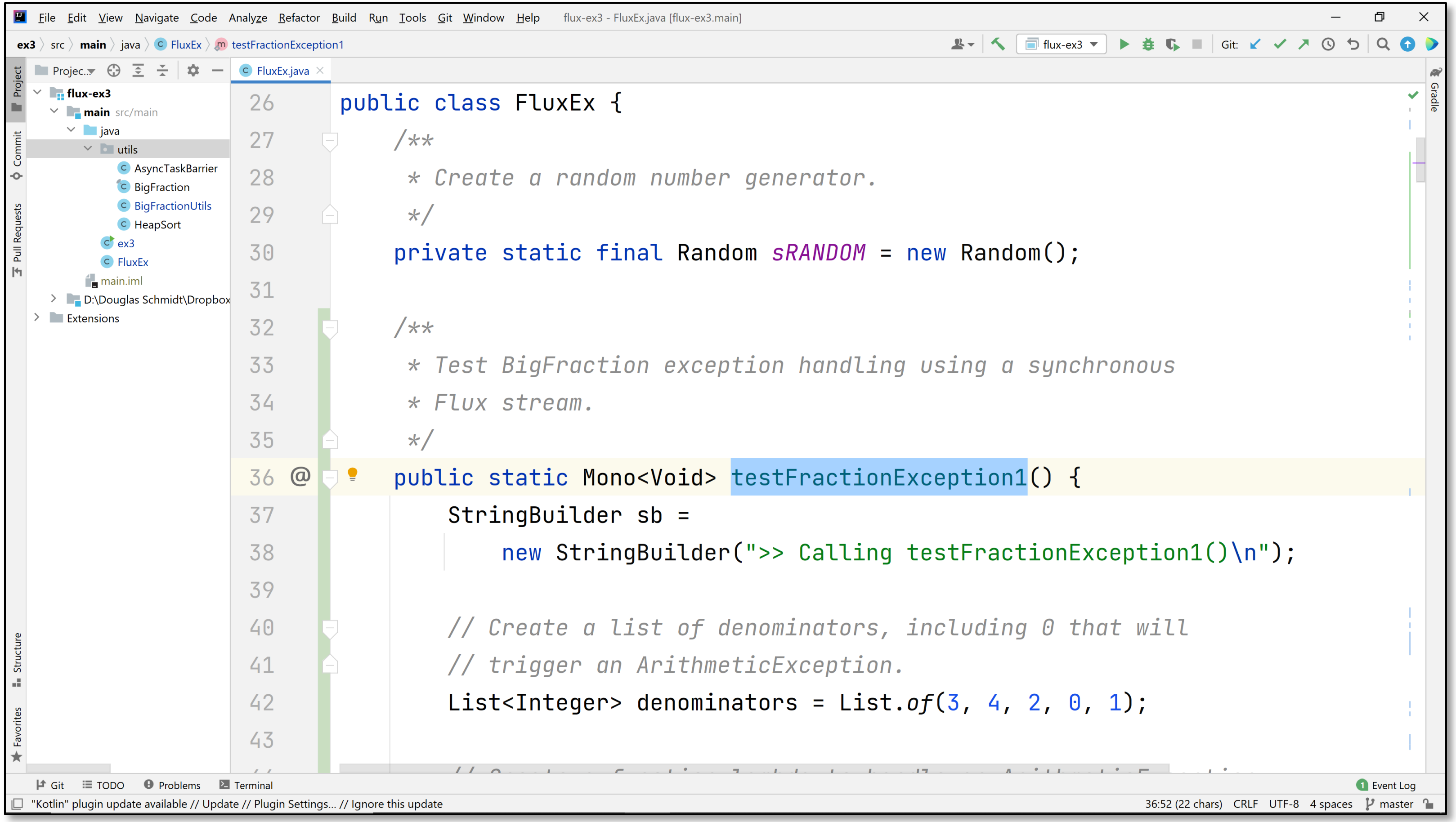
- Part 1 of case study ex3 shows how to use Flux operators fromIterable(), flatMap(), map(), onErrorResume(), onErrorStop(), collectList(), filter(), onErrorContinue(), & the parallel thread pool to create, reduce, multiply, & display BigFraction objects (a)synchronously
  - It also shows the use of Mono operators like fromCallable(), subscribeOn(), firstWithSignal(), flatMap(), onErrorResume(), then(), & doOnSuccess()

```
Mono<List<BigFraction>> qSortM =
 Mono.fromCallable(() ->
               quickSort(list))
  .subscribeOn
    (Schedulers.parallel());
Mono<List<BigFraction>> hSortM =
  Mono.fromCallable(() ->
               heapSort(list))
  .subscribeOn
    (Schedulers.parallel());

return Mono.firstWithSignal
             (qSortM, hSortM)
   .doOnSuccess(displayList)
   .then();
```

Applying Key Operators in the Flux Class to ex3

Applying Key Operators in the Flux Class to ex3


File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Help
flux-ex3 - FluxEx.java [flux-ex3.main]
ex3 〉 src 〉 main 〉 java 〉 FluxEx 〉 testFractionException1
flux-ex3
Git:
Projec...
FluxEx.java
flux-ex3
main src/main
java
utils
AsyncTaskBarrier
BigFraction
BigFractionUtils
HeapSort
ex3
FluxEx
main.iml
D:\Douglas Schmidt\Dropbox
Extensions
Project
Commit
Pull Requests
Structure
Favorites
Gradle
Git
TODO
Problems
Terminal
Event Log
"Kotlin" plugin update available // Update // Plugin Settings... // Ignore this update
36:52 (22 chars) CRLF UTF-8 4 spaces master

```java
public class FluxEx {
    /**
     * Create a random number generator.
     */
    private static final Random sRANDOM = new Random();

    /**
     * Test BigFraction exception handling using a synchronous
     * Flux stream.
     */
    public static Mono<Void> testFractionException1() {
        StringBuilder sb =
            new StringBuilder(">> Calling testFractionException1()\n");

        // Create a list of denominators, including 0 that will
        // trigger an ArithmeticException.
        List<Integer> denominators = List.of(3, 4, 2, 0, 1);

```

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

End of Applying Key Methods in the Flux Class: Case Study ex3 (Part 1)