

Applying Key Operators in the Mono Class: Case Study ex2 (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 ex2 applies the Mono operators fromCallable(), map(), then(), doOnSuccess(), just(), blockOptional(), onErrorResume(), subscribeOn(), & Schedulers.single() to create, reduce, multiply, & display BigFraction objects asynchronously

```
    return Mono
        .fromCallable(call)
        .subscribeOn
            (Schedulers.single())
        .onErrorResume(errorHandler)
        .doOnSuccess(printResult)
        .then();
```

Learning Objectives in this Part of the Lesson

- Part 2 of case study ex2 applies the Mono operators fromCallable(), map(), then(), doOnSuccess(), just(), blockOptional(), onErrorResume(), subscribeOn(), & Schedulers.single() to create, reduce, multiply, & display BigFraction objects asynchronously
 - An operator for handling runtime exceptions is shown

```
return Mono
    .fromCallable(call)
    .subscribeOn
        (Schedulers.single())
    .onErrorResume(errorHandler)
    .doOnSuccess(printResult)
    .then();
```

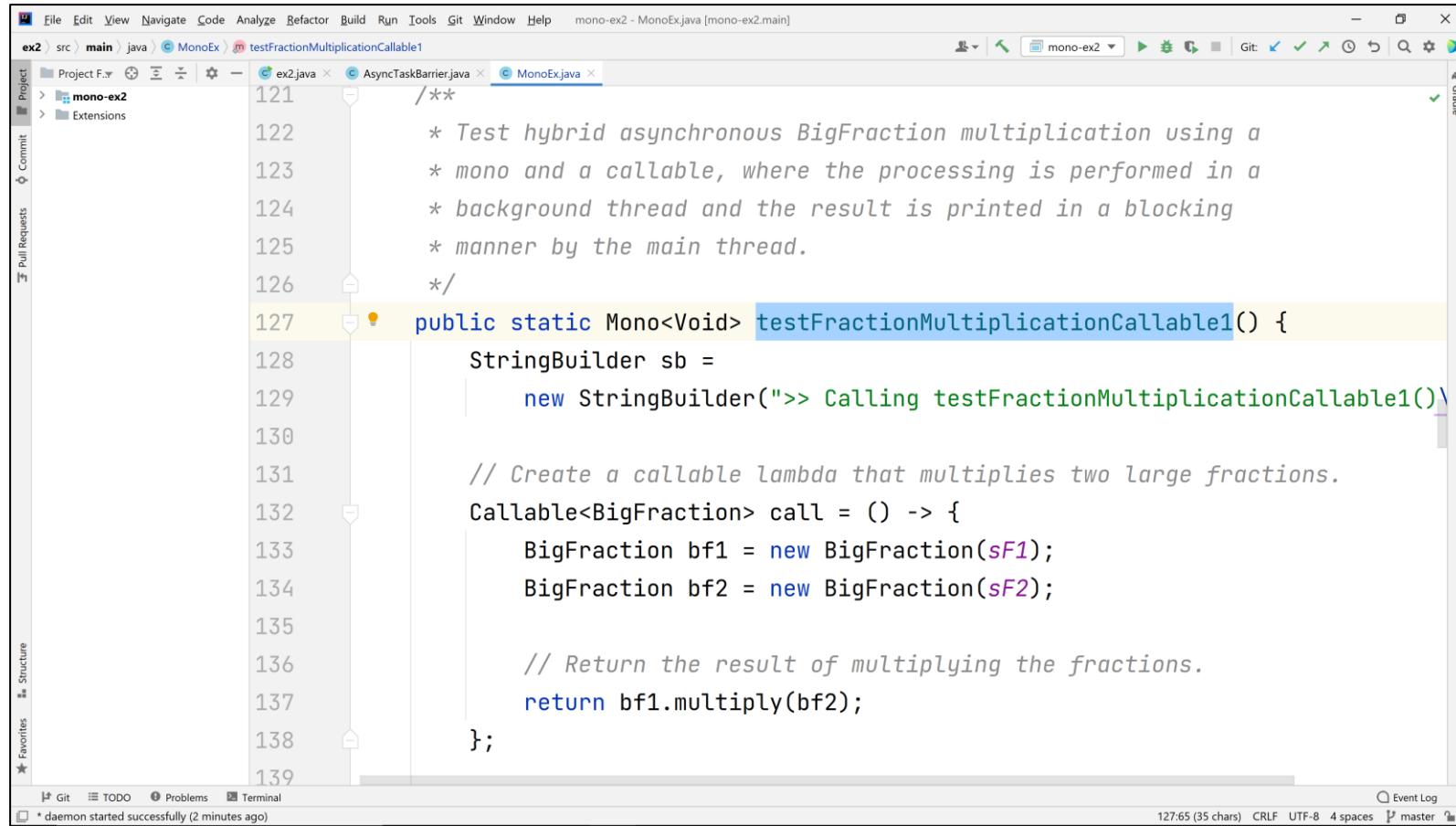
Learning Objectives in this Part of the Lesson

- Part 2 of case study ex2 applies the Mono operators fromCallable(), map(), then(), doOnSuccess(), just(), blockOptional(), onErrorResume(), subscribeOn(), & Schedulers.single() to create, reduce, multiply, & display BigFraction objects asynchronously
 - An operator for handling runtime exceptions is shown
 - Both fully asynchronous & hybrid asynchronous/synchronous models are also shown

```
Optional<BigFraction> result =  
    Mono  
        .fromCallable(call)  
        .subscribeOn  
            (Schedulers.single())  
        .blockOptional();  
    ...
```

Applying Key Methods in the Mono Class in ex2

Applying Key Methods in the Mono Class in ex2



The screenshot shows an IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, Git, Window, Help.
- Project:** mono-ex2 - MonoEx.java [mono-ex2.main].
- Editor:** ex2 src main java MonoEx testFractionMultiplicationCallable1.
- Code:** ex2.java, AsyncTaskBarrier.java, MonoExJava (selected).
- Content:** The code implements a `Mono<Void>` for testing fraction multiplication. It uses a `StringBuilder` to build a log message, creates a `Callable<BigFraction>` lambda that multiplies two `BigFraction` objects, and returns the result of the multiplication.

```
121     * Test hybrid asynchronous BigFraction multiplication using a
122     * mono and a callable, where the processing is performed in a
123     * background thread and the result is printed in a blocking
124     * manner by the main thread.
125     */
126
127     public static Mono<Void> testFractionMultiplicationCallable1() {
128         StringBuilder sb =
129             new StringBuilder(">> Calling testFractionMultiplicationCallable1()\n");
130
131         // Create a callable lambda that multiplies two large fractions.
132         Callable<BigFraction> call = () -> {
133             BigFraction bf1 = new BigFraction(sF1);
134             BigFraction bf2 = new BigFraction(sF2);
135
136             // Return the result of multiplying the fractions.
137             return bf1.multiply(bf2);
138         };
139     }
```

- Bottom Status Bar:** Git, TODO, Problems, Terminal, Event Log, daemon started successfully (2 minutes ago), 127:65 (35 chars), CRLF, UTF-8, 4 spaces, master.

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

End of Applying Key Operators in the Mono Class: Case Study ex2 (Part 2)