

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 from `Callable()`, `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();
```

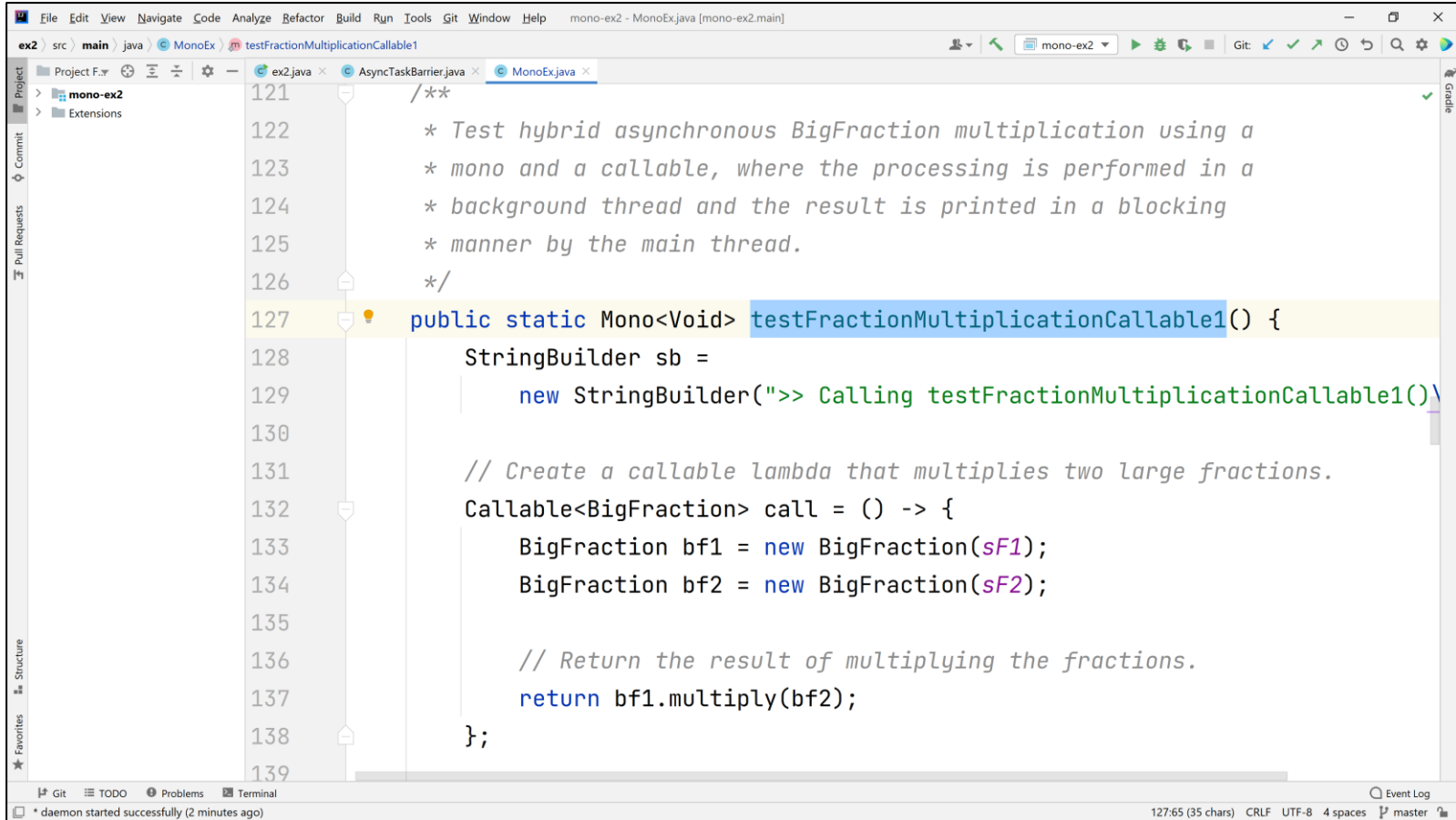
Learning Objectives in this Part of the Lesson

- 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



```
121  /**
122   * Test hybrid asynchronous BigFraction multiplication using a
123   * mono and a callable, where the processing is performed in a
124   * background thread and the result is printed in a blocking
125   * manner by the main thread.
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  }
```

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

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