

Key Transforming Operators in the Flux Class (Part 2)

Douglas C. Schmidt

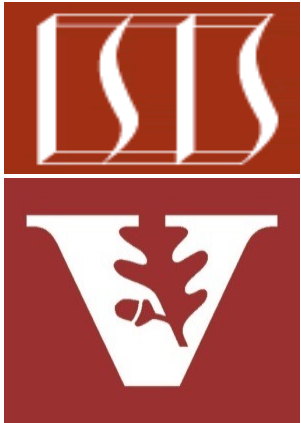
d.schmidt@vanderbilt.edu

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

- Recognize key Flux operators
 - Factory method operators
- Transforming operators
 - Transform the values and/or types emitted by a Flux
 - e.g., flatMap()



Key Transforming Operators in the Flux Class

Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously

```
<R> Flux<R> flatMap  
(Function<? super T,  
? extends Publisher<?  
extends R>>  
mapper)
```

Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers

```
<R> Flux<R> flatMap  
(Function<? super T,  
? extends Publisher<?  
extends R>>  
mapper)
```

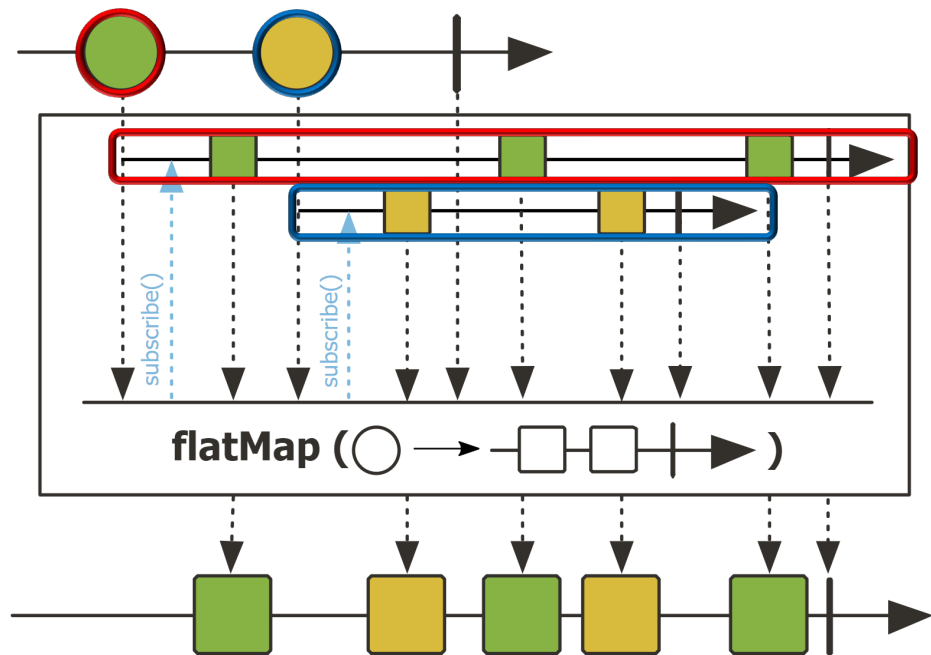
Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Each <T> input element is mapped to a Publisher<R>

```
<R> Flux<R> flatMap  
(Function<? super T,  
? extends Publisher<?  
extends R>>  
mapper)
```

Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Each $\langle T \rangle$ input element is mapped to a Publisher $\langle R \rangle$
 - That Publisher will emit zero or more items



Key Transforming Operators in the Flux Class

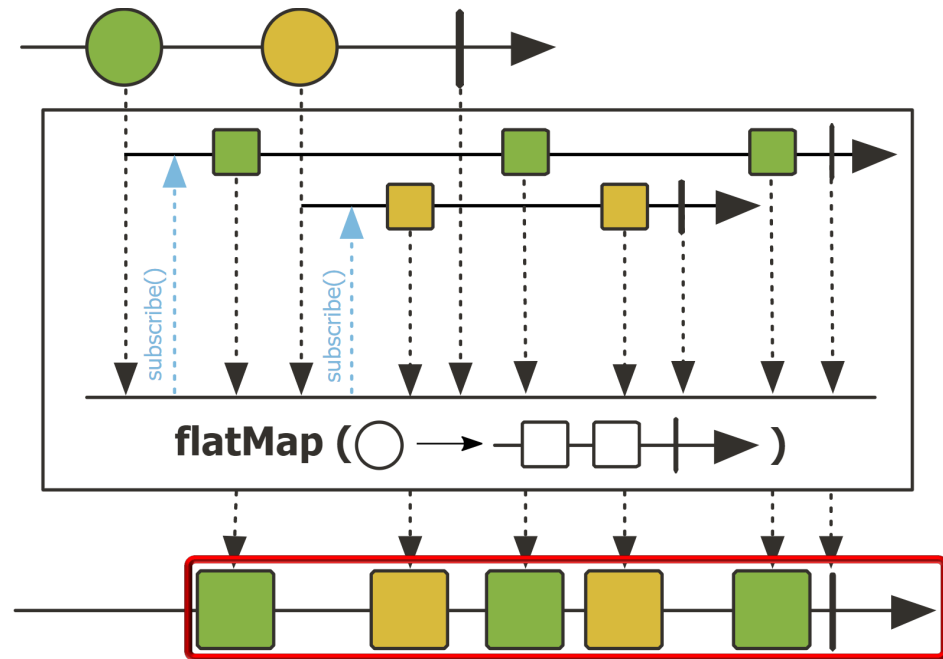
- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Inner publishers are “flattened” into one Flux by merging

```
<R> Flux<R> flatMap  
(Function<? super T,  
? extends Publisher<?  
extends R>>  
mapper)
```



Key Transforming Operators in the Flux Class

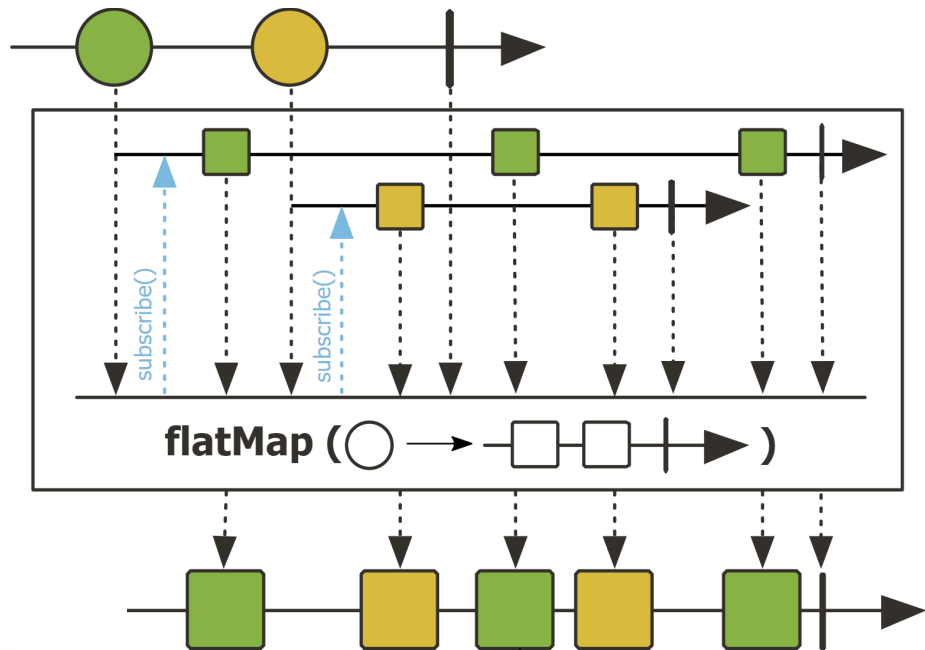
- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Inner publishers are “flattened” into one Flux by merging
 - They can therefore interleave
 - Especially when used for concurrent processing



See upcoming walkthrough of the “flatMap() concurrency idiom” example

Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Inner publishers are “flattened” into one Flux by merging
 - It has similarities & differences compared to map()

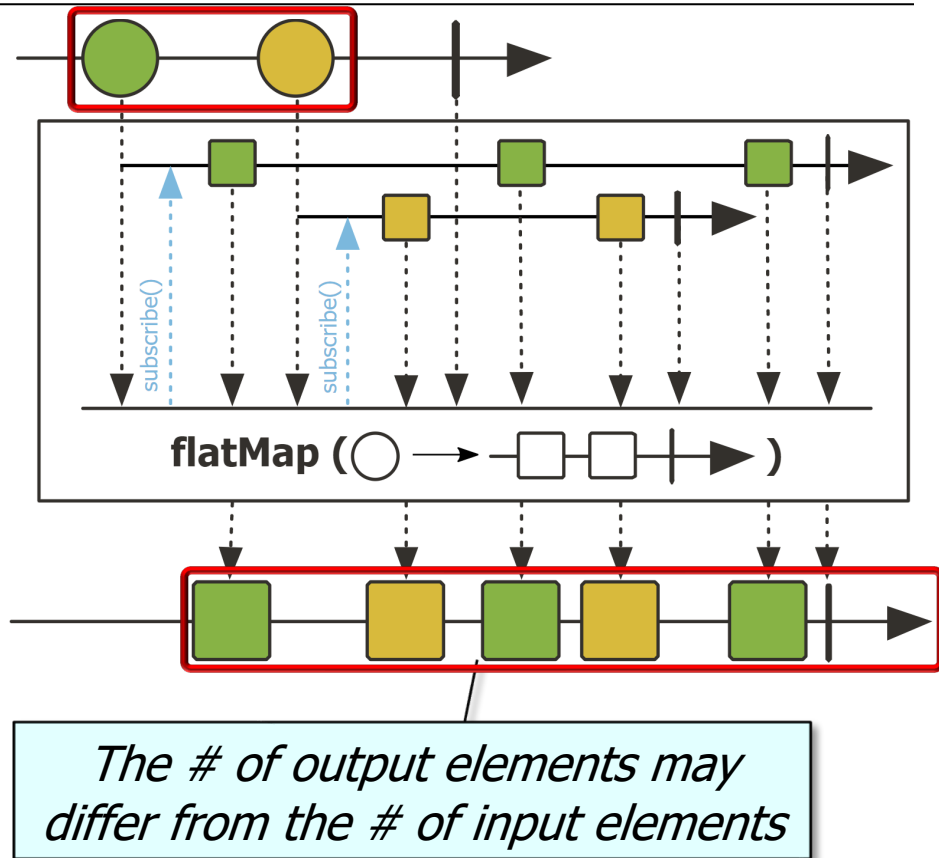


flatMap() can transform the values and/or type of elements it processes



Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - These elements are emitted into inner Publishers
 - Inner publishers are “flattened” into one Flux by merging
 - It has similarities & differences compared to map()



Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing



```
return Flux
    .fromIterable(denominators)

    .flatMap(denominator -> Mono
        .fromCallable(() ->
            BigFraction
                .valueOf(...,
                    denominator))

        .subscribeOn
            (Schedulers
                .parallel())

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

See upcoming discussion on the Project Reactor flatMap() concurrency idiom

Key Transforming Operators in the Flux Class

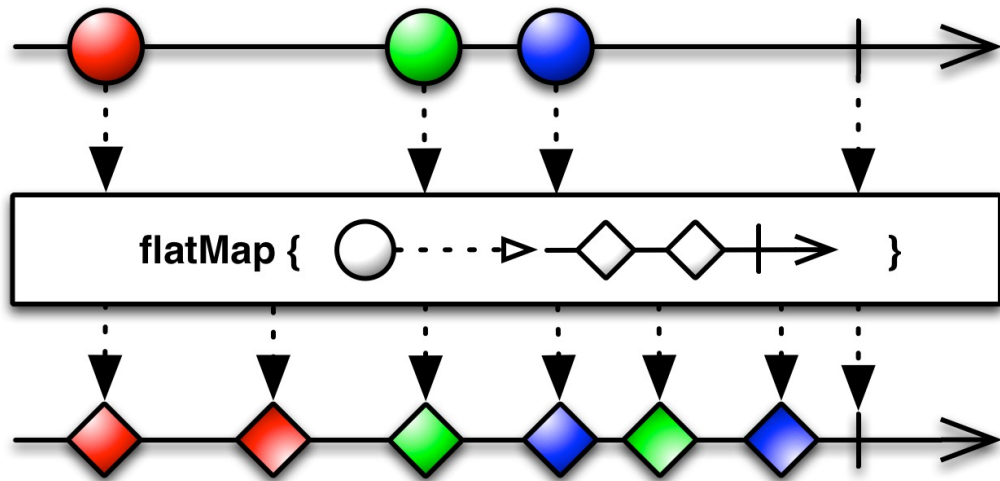
- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing

```
return Flux
    .fromIterable(denominators)
    .flatMap(denominator -> Mono
        .fromCallable(() ->
            BigFraction
                .valueOf(...,
                    denominator))
        .subscribeOn
            (Schedulers
                .parallel())
        .map(bf -> bf
            multiply(sBigFrac)))
```

Return a Flux to a multiplied big fraction using the Project Reactor "flatMap() concurrency idiom"

Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing
 - RxJava's Observable.flatMap() operator works the same way



Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing
 - RxJava's Observable.flatMap() operator works the same way
 - Similar to the Java Streams flatMap() operation

flatMap

```
<R> Stream<R> flatMap(  
Function<? super T,? extends Stream<? extends R>> mapper)
```

Returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. Each mapped stream is closed after its contents have been placed into this stream. (If a mapped stream is null an empty stream is used, instead.)

```
List<String> a = List.of("d", "g");  
List<String> b = List.of("a", "c");  
Stream  
    .of(a, b)  
    .flatMap(List::stream)  
    .sorted()  
    .forEach(System.out::println);
```

*Flatten, sort, & print
two lists of strings*

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#flatMap

Key Transforming Operators in the Flux Class

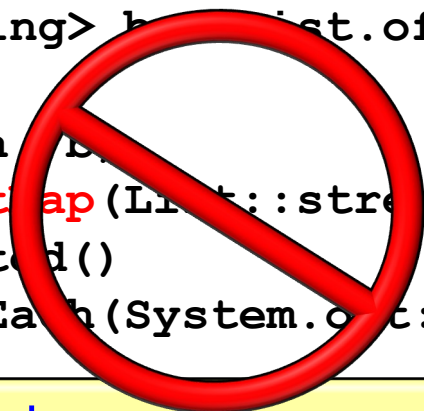
- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing
 - RxJava's Observable.flatMap() operator works the same way
- Similar to the Java Streams flatMap() operation
 - However, Stream.flatMap() doesn't support parallelism..

flatMap

```
<R> Stream<R> flatMap(  
Function<? super T,? extends Stream<? extends R>> mapper)
```

Returns a stream consisting of the results of replacing each element of this stream with the contents of a mapped stream produced by applying the provided mapping function to each element. Each mapped stream is closed after its contents have been placed into this stream. (If a mapped stream is null an empty stream is used, instead.)

```
List<String> a = List.of("d", "g");  
List<String> b = List.of("a", "c");  
Stream  
    .of(a, b)  
    .flatMap(List::stream)  
    .sorted()  
    .forEach(System.out::println);
```



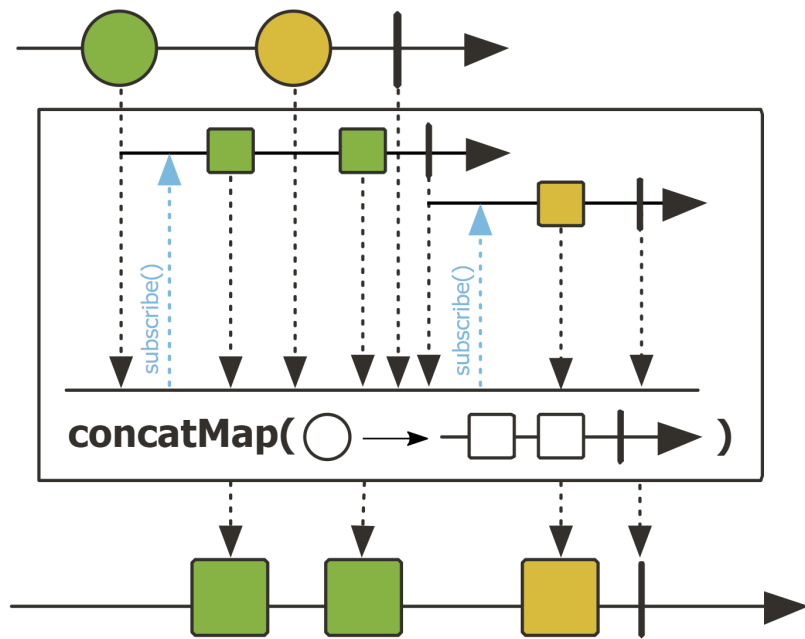
Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a)synchronously
 - This method is often used to trigger concurrent processing
 - RxJava's Observable.flatMap() operator works the same way
 - Similar to the Java Streams flatMap() operation
 - flatMap() doesn't ensure the order of the items in the resulting stream



Key Transforming Operators in the Flux Class

- The flatMap() operator
 - Transform the elements emitted by this Flux (a) synchronously
 - This method is often used to trigger concurrent processing
 - RxJava's Observable.flatMap() operator works the same way
 - Similar to the Java Streams flatMap() operation
 - flatMap() doesn't ensure the order of the items in the resulting stream
 - Use concatMap() if order matters



End of Key Transforming Operators in the Flux Class (Part 2)