

# Key Transforming Operators in the Flux Class (Part 1)

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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., `map()` & `mapNotNull()`



---

# Key Transforming Operators in the Flux Class

# Key Transforming Operators in the Flux Class

---

- The map() operator
  - Transform the item(s) emitted by this Flux

```
<V> Flux<V> map  
(Function<? super T, ? extends V>  
 mapper)
```

# Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item

```
<V> Flux<V> map  
(Function<? super T, ? extends V>  
 mapper)
```

## Interface Function<T,R>

### Type Parameters:

T - the type of the input to the function

R - the type of the result of the function

### All Known Subinterfaces:

UnaryOperator<T>

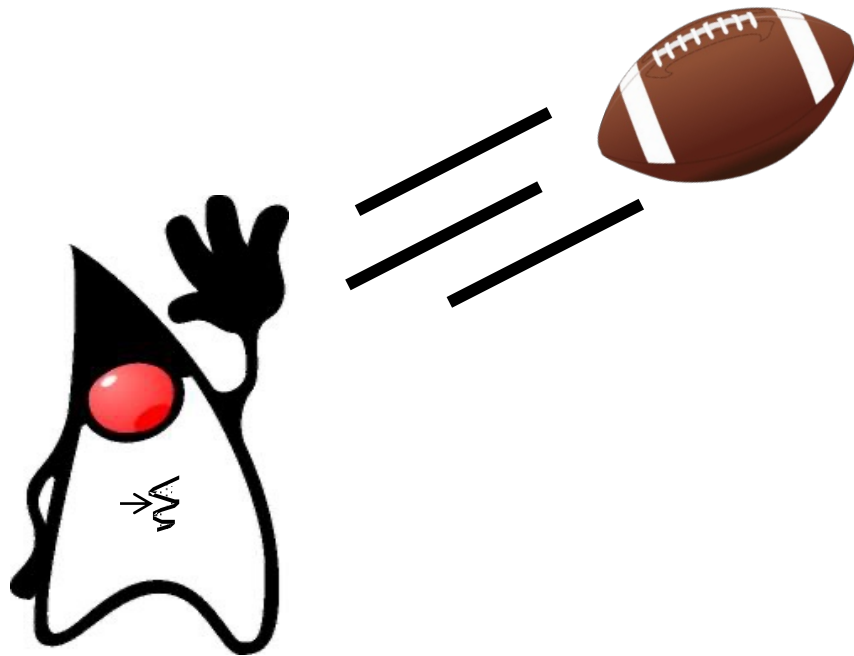
### Functional Interface:

This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

# Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
    - Applies a synchronous function to transform each item
    - map() can terminate if mapper throws an exception

```
<V> Flux<V> map  
(Function<? super T, ? extends V>  
 mapper)
```



# Key Transforming Operators in the Flux Class

---

- The map() operator
  - Transform the item(s) emitted by this Flux
    - Applies a synchronous function to transform each item
  - Returns a transformed Flux

```
<V> Flux<V> map  
(Function<? super T, ? extends V>  
 mapper)
```



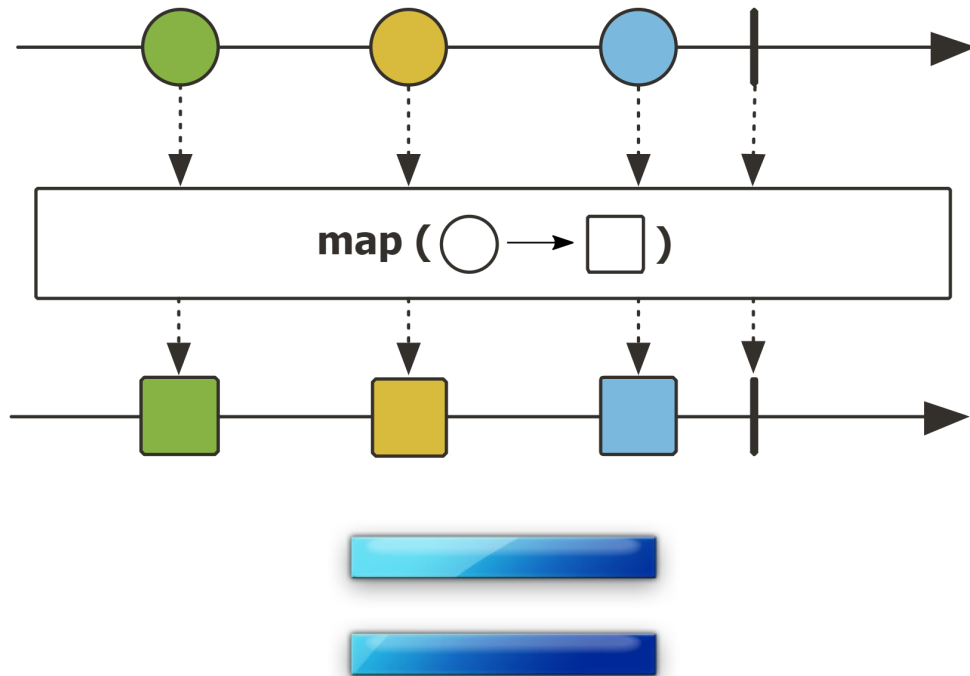
# Key Transforming Operators in the Flux Class

- The `map()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items

**Flux**

```
.fromIterable  
  (bigFractionList)  
...  
.map(fraction -> fraction  
    .multiply(sBigReducedFrac))  
...
```

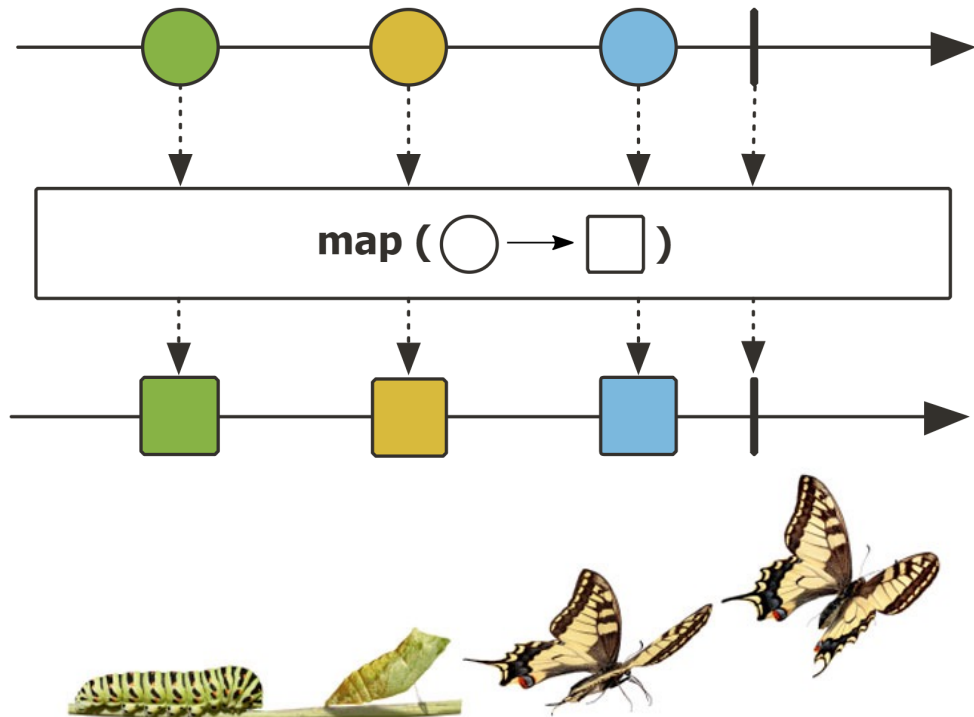
*Multiply each element in the Flux stream by a constant*



See [Reactive/flux/ex1/src/main/java/FluxEx.java](https://github.com/reactive/reactive-streams-examples/blob/master/flux-examples/src/main/java/FluxEx.java)

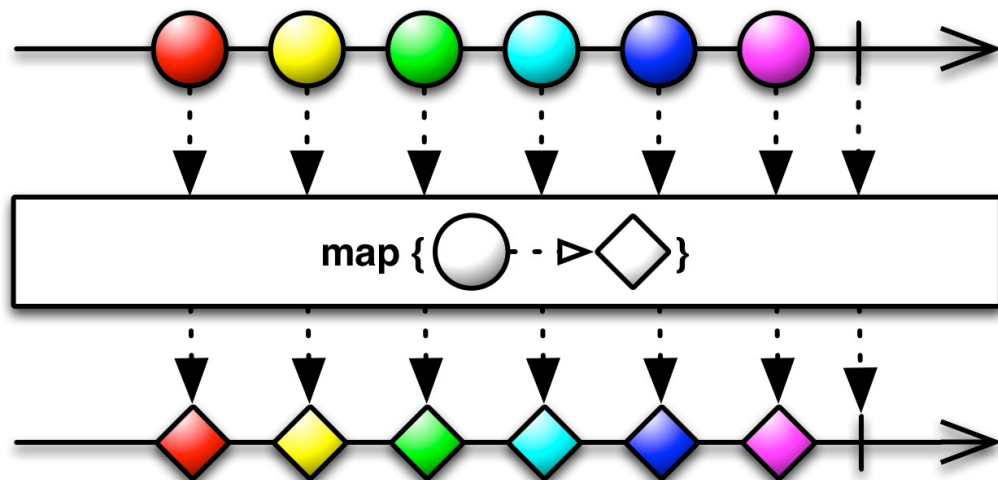
# Key Transforming Operators in the Flux Class

- The `map()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
  - `map()` can transform the type and/or value of elements it processes



# Key Transforming Operators in the Flux Class

- The `map()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava's `Observable.map()` operator works the same



Observable

```
.fromIterable(bigFractionList)
...
.map(fraction -> fraction
    .multiply(sBigReducedFrac))
...
```

*Multiply each element in the Observable stream by a constant*

# Key Transforming Operators in the Flux Class

- The map() operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
  - RxJava's Observable.map() operator works the same
  - Similar to Stream.map() in Java Streams

```
List<String> collect = List
    .of("a", "b", "c").stream()
    .map(String::toUpperCase).collect(toList());
```

*Uppercase strings &  
collect into a List*

## map

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

Returns a stream consisting of the results of applying the given function to the elements of this stream.

This is an intermediate operation.

### Type Parameters:

R - The element type of the new stream

### Parameters:

mapper - a non-interfering, stateless function to apply to each element

# Key Transforming Operators in the Flux Class

---

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
 mapper)
```

# Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
mapper)
```

## Interface Function<T,R>

### Type Parameters:

T - the type of the input to the function

R - the type of the result of the function

### All Known Subinterfaces:

UnaryOperator<T>

### Functional Interface:

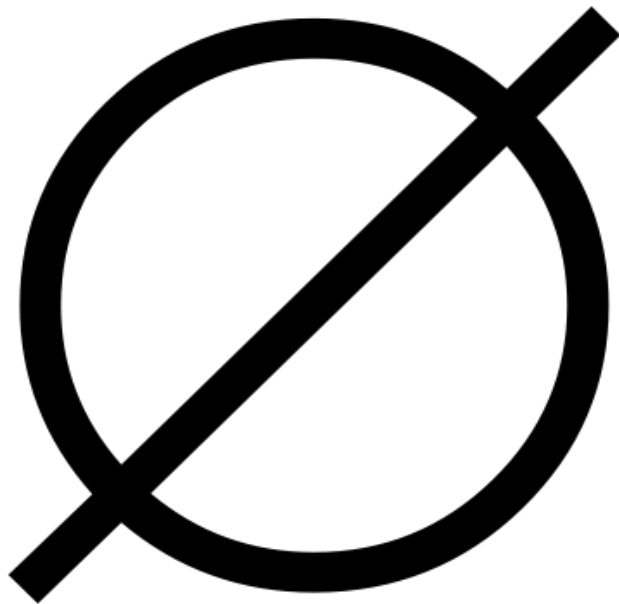
This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

# Key Transforming Operators in the Flux Class

---

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
  - It's possible for a mapper function to produce null values

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
mapper)
```



# Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
  - It's possible for a mapper function to produce null values
    - However, these null values are not emitted

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
 mapper)
```



# Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
  - It's possible for a mapper function to produce null values
    - These null values are not emitted
    - Behaves like map(Function) followed by filter(Predicate)
      - However, null is not a supported value, so it can't be filtered out

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
 mapper)
```

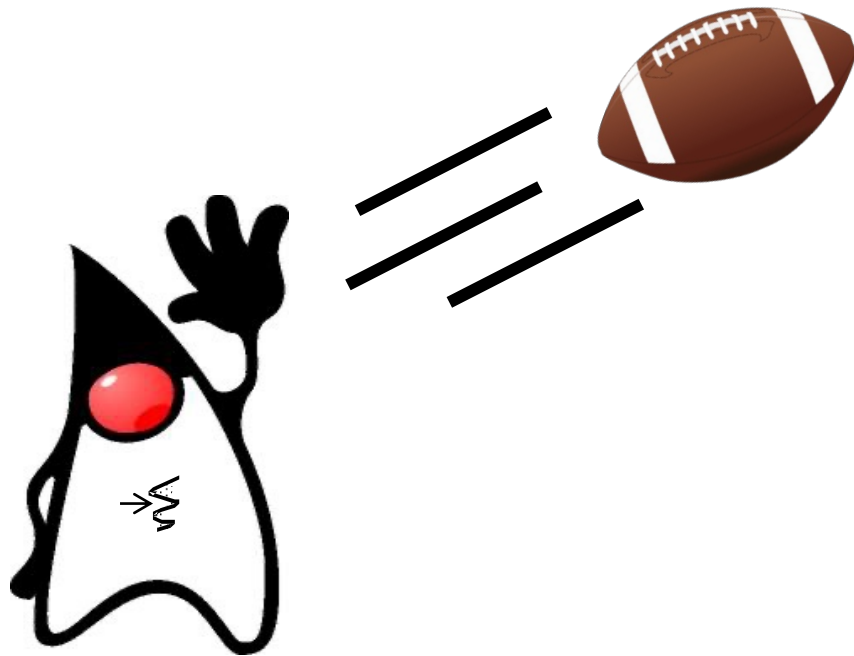


```
Flux  
...  
.map(this.mapper)  
.filter(v -> v != null)  
...
```

# Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
  - Applies a synchronous function to transform each item
    - It's possible for a mapper function to produce null values
  - mapNotNull() can terminate if mapper throws an exception

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
 mapper)
```



# Key Transforming Operators in the Flux Class

- The mapNotNull() operator
  - Transform the item(s) emitted by this Flux
    - Applies a synchronous function to transform each item
  - Returns a transformed Flux that emits no nulls

```
<V> Flux<V> mapNotNull  
(Function<? super T, ? extends V>  
 mapper)
```

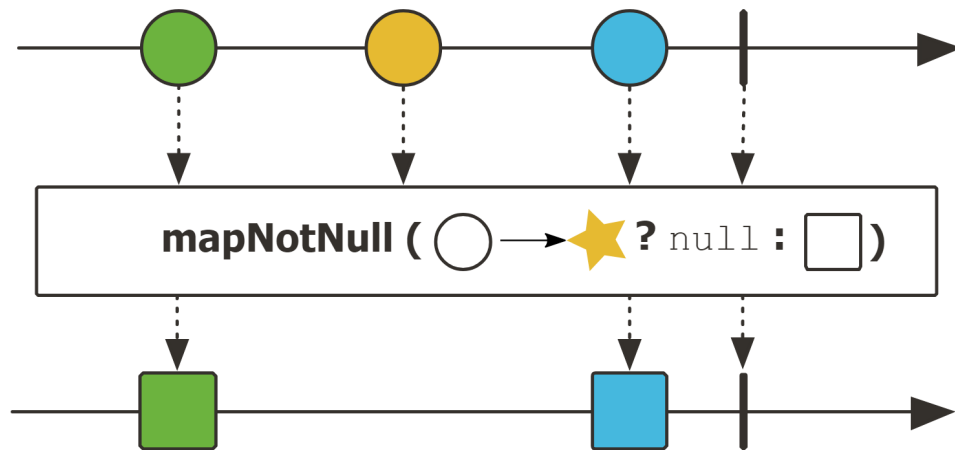


# Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items

Flux

```
.just(BigFraction  
    .valueOf(100, 3),  
    BigFraction  
    .valueOf(100, 10))  
.mapNotNull(bf -> bf  
    .equals(BigFraction.TEN)  
    ? null : bf)  
...
```

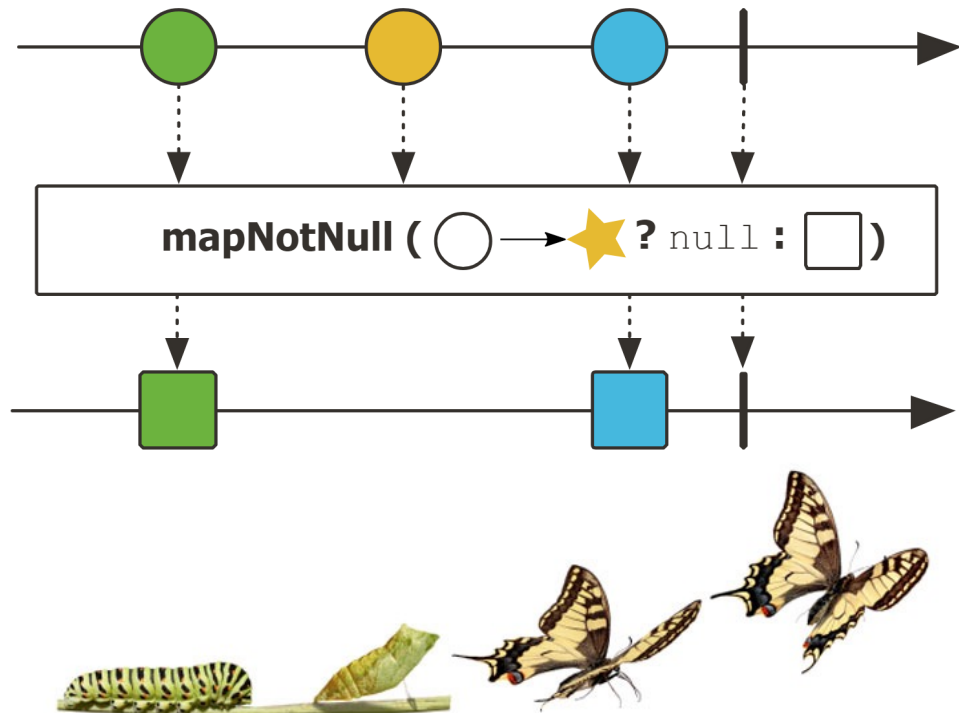


*Return null if bf equals 10, which is then ignored*

See [Reactive/flux/ex1/src/main/java/FluxEx.java](https://github.com/reactive/reactive-streams-examples/blob/master/java/flux-ex/src/main/java/FluxEx.java)

# Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
  - `mapNotNull()` can transform the type and/or value of elements it processes



# Key Transforming Operators in the Flux Class

---

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava's Observable lacks a `mapNotNull()` operator



---

See [github.com/ReactiveX/RxJava/issues/4644](https://github.com/ReactiveX/RxJava/issues/4644)

# Key Transforming Operators in the Flux Class

---

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava's Observable lacks a `mapNotNull()` operator
  - Java Optional can be used in this case

```
return Observable
    .fromCallable(() -> url)

    .subscribeOn(Schedulers.io())

    .map(__ -> Optional
        .ofNullable(download(url)))

    .filter(Optional::isPresent)

    .map(Optional::get);
```

# Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava's Observable lacks a `mapNotNull()` operator
  - Java Optional can be used in this case
  - RxJava transformers can also be used

```
static <T, R>
ObservableTransformer<T, R>
mapNotNull(Function<? super T,
              ? extends R>
              mapper) {
    return upstream -> upstream
        .flatMap(it -> {
            R result = mapper.apply(it);
            if (result == null)
                return Observable.empty();
            else
                return Observable
                    .just(result);
        });
}
```

# Key Transforming Operators in the Flux Class

- The `mapNotNull()` operator
  - Transform the item(s) emitted by this Flux
  - The # of output items must match the # of input items
- RxJava's Observable lacks a `mapNotNull()` operator
  - Java Optional can be used in this case
  - RxJava transformers can also be used

```
Observable<Image> downloadImage
(URL url) {
    return Observable
        .fromCallable(() -> url)

        .subscribeOn(Schedulers.io())

        .compose(mapNotNull
            (this::download));
}
```

---

# End of Key Transforming Operators in the Flux Class (Part 1)