Key Factory Method Operators in the Flux Class (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

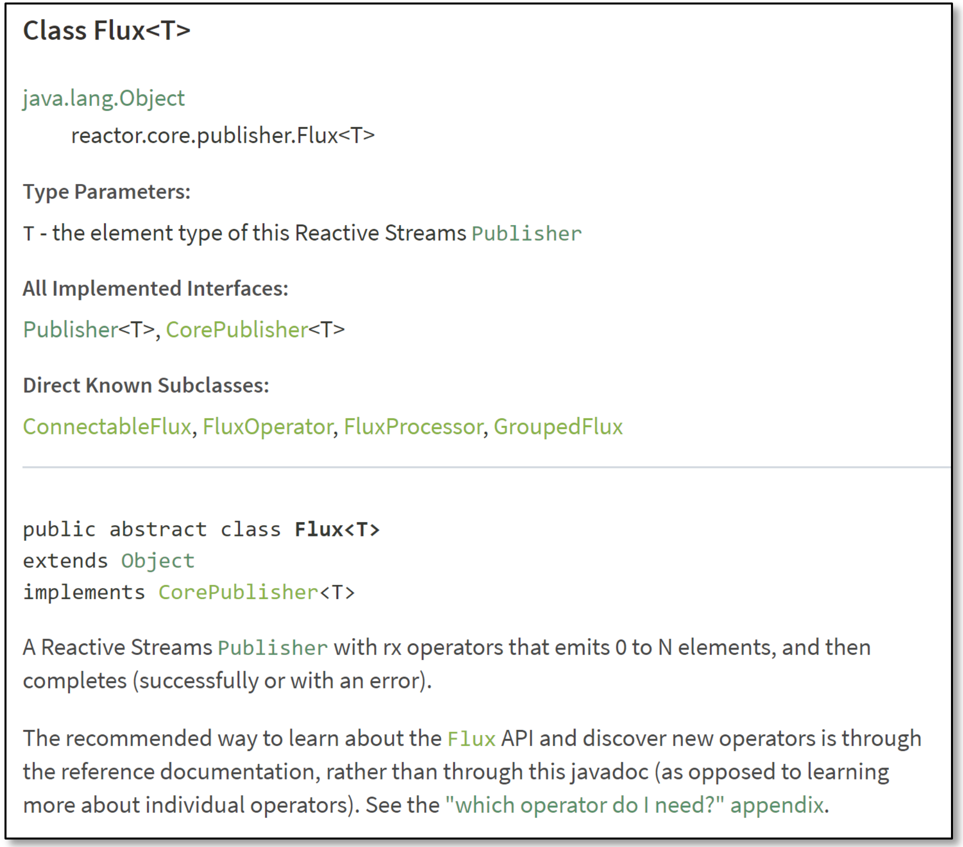
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Recognize key Flux operators



Class Flux<T>

java.lang.Object
    reactor.core.publisher.Flux<T>

Type Parameters:

T - the element type of this Reactive Streams Publisher

All Implemented Interfaces:

Publisher<T>, CorePublisher<T>

Direct Known Subclasses:

ConnectableFlux, FluxOperator, FluxProcessor, GroupedFlux

```
public abstract class Flux<T>
extends Object
implements CorePublisher<T>
```

A Reactive Streams Publisher with rx operators that emits 0 to N elements, and then completes (successfully or with an error).

The recommended way to learn about the Flux API and discover new operators is through the reference documentation, rather than through this javadoc (as opposed to learning more about individual operators). See the "which operator do I need?" appendix.

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html

Learning Objectives in this Part of the Lesson

- Recognize key Flux operators
  - Factory method operators
    - These operators create Flux streams in various ways
      - e.g., just(), fromArray(), fromIterable(), & from()

See en.wikipedia.org/wiki/Factory_method_pattern

Key Factory Method Operators in the Flux Class

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes

```
static <T> Flux<T> just(T... data)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#just

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit
      - Passed as a vararg

```
static <T> Flux<T> just(T... data)
```

```
var jenny = Flux.just(8);
var jenny = Flux.just(8,6);
var jenny = Flux.just(8,6,7);
var jenny = Flux.just(8,6,7,5);
var jenny = Flux.just(8,6,7,5,3);
var jenny = Flux.just(8,6,7,5,3,0);
var jenny = Flux.just(8,6,7,5,3,0,9);
```

See www.baeldung.com/java-varargs

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit
    - Returns a new Flux that's captured at "assembly time"
      - i.e., it's "eager"

```
static <T> Flux<T> just(T... data)
```

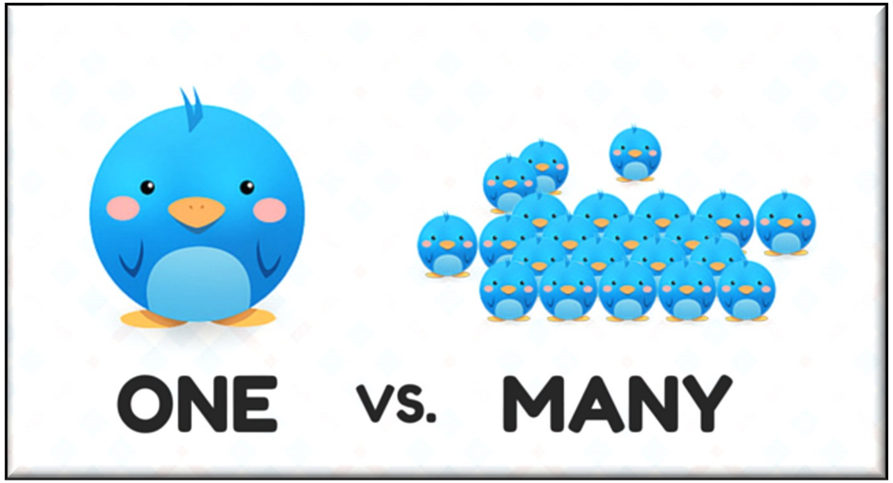
"Assembly time" is when the Flux object is instantiated, rather than when it "runs"

Key Factory Method Operators in the Flux Class

- The just() operator

```
static <T> Flux<T> just(T... data)
```

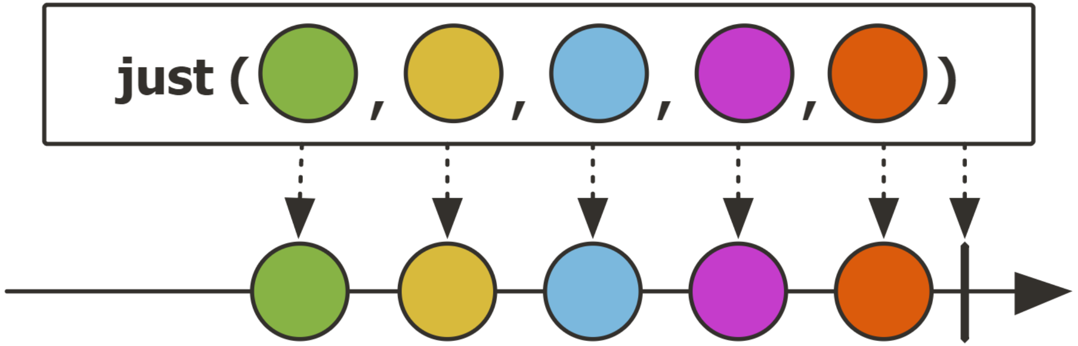
  - Create a Flux that emits the given element(s) & then completes
    - The param(s) are the elements to emit
    - Returns a new Flux that's captured at instantiation time
    - Multiple elements can be emitted, unlike the Mono.just() operator



See projectreactor.io/docs/core/release/api/reactor/core/publisher/Mono.html#just

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model



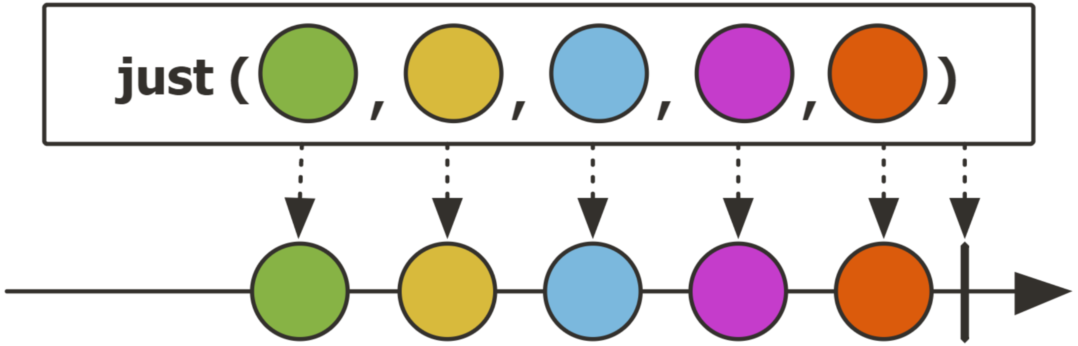
```
Flux
  .just(BigFraction.valueOf(100,3),
        BigFraction.valueOf(100,4),
        BigFraction.valueOf(100,2),
        BigFraction.valueOf(100,1))
  ...
```

Create a Flux stream of four BigFraction objects

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Factory Method Operators in the Flux Class

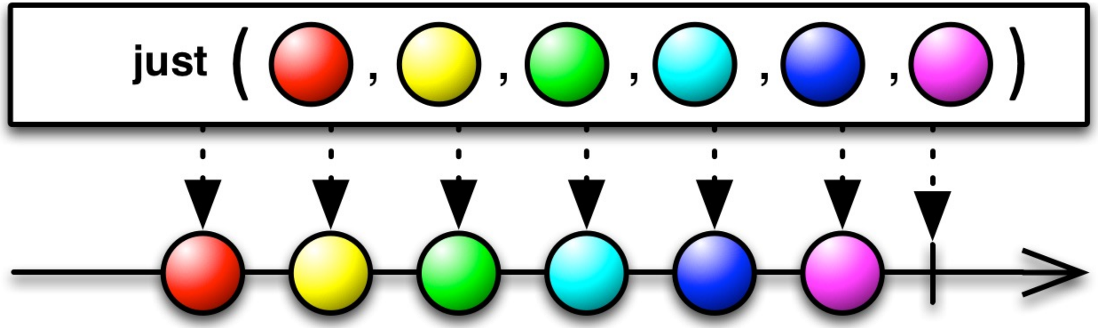
- The just() operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model
    - Since just() is evaluated eagerly at "assembly time" it runs in the thread where assembly is performed



The fromIterable() & fromArray() factory method operators also evaluate eagerly

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model
  - RxJava's Observable.just() works the same



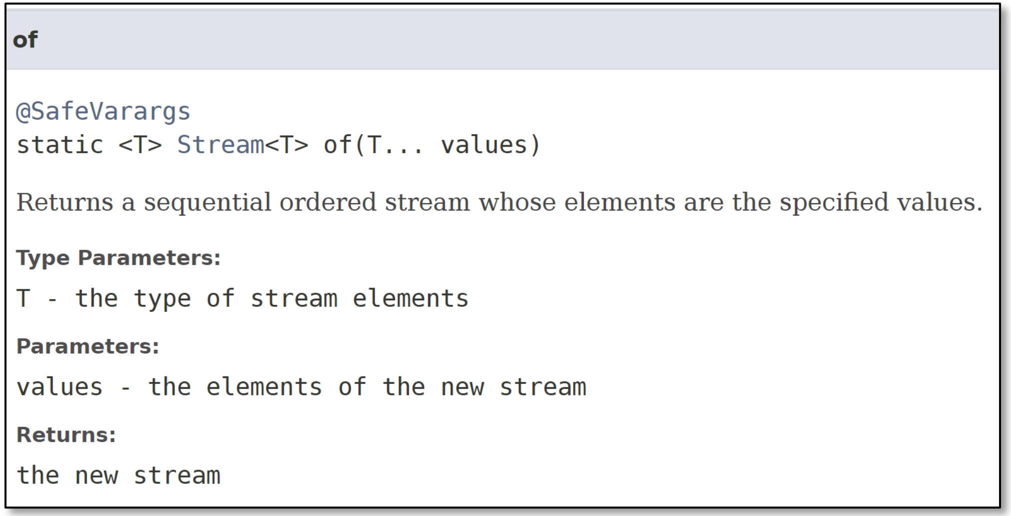
```
Observable
    .just(BigFraction.valueOf(100,3),
          BigFraction.valueOf(100,4),
          BigFraction.valueOf(100,2),
          BigFraction.valueOf(100,1))
    ...
```

Create an Observable stream of four BigFraction objects

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#just

Key Factory Method Operators in the Flux Class

- The just() operator
  - Create a Flux that emits the given element(s) & then completes
  - This factory method operator adapts non-reactive input sources to the reactive model
  - RxJava's Observable.just() works the same
  - Similar to the Stream.of() operator in Java Streams



of

```
@SafeVarargs
static <T> Stream<T> of(T... values)
```

Returns a sequential ordered stream whose elements are the specified values.

Type Parameters:

`T - the type of stream elements`

Parameters:

`values - the elements of the new stream`

Returns:

`the new stream`

```
Stream
  .of(BigFraction.valueOf(100,3),
      BigFraction.valueOf(100,4),
      BigFraction.valueOf(100,2),
      BigFraction.valueOf(100,1))
  ...
```

Create a stream of 4 BigFraction objects

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

Key Factory Method Operators in the Flux Class

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable

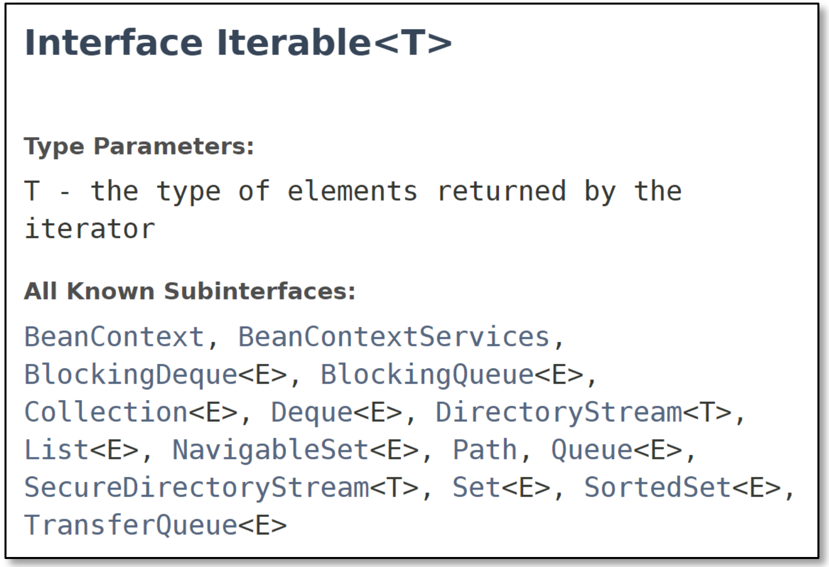
```
static <T> Flux<T> fromIterable
   (Iterable<? extends T> it)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#fromIterable

Key Factory Method Operators in the Flux Class

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable
    - The Iterable.iterator() method will be invoked at least once & at most twice for each subscriber

```
static <T> Flux<T> fromIterable
    (Iterable<? extends T> it)
```



Interface Iterable<T>

Type Parameters:

T - the type of elements returned by the iterator

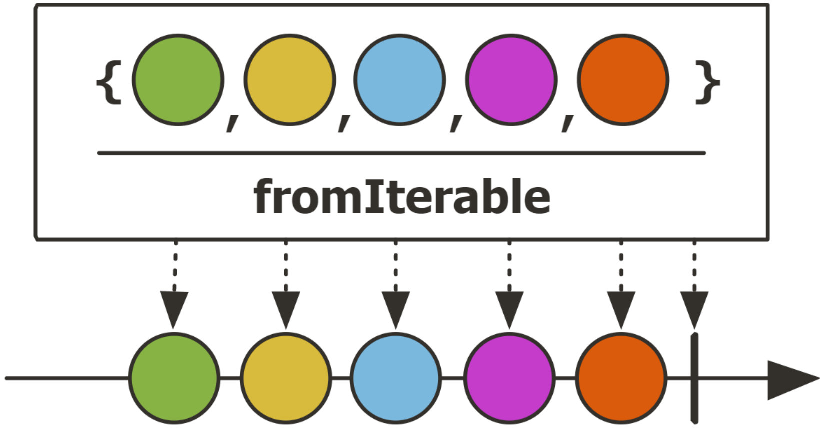
All Known Subinterfaces:

BeanContext, BeanContextServices, BlockingDeque<E>, BlockingQueue<E>, Collection<E>, Deque<E>, DirectoryStream<T>, List<E>, NavigableSet<E>, Path, Queue<E>, SecureDirectoryStream<T>, Set<E>, SortedSet<E>, TransferQueue<E>

See docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html

Key Factory Method Operators in the Flux Class

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable
  - This factory method operator also adapts non-reactive input sources into the reactive model
    - e.g., Java collections like List & Set



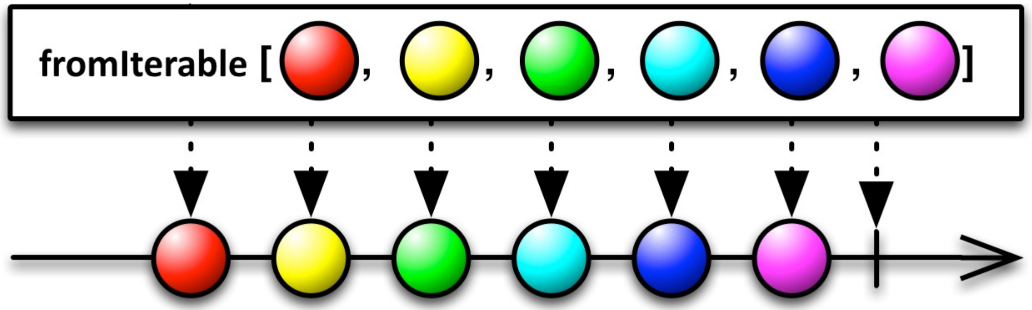
```
List<Integer> list =
  List.of(0,1,1,2,3,5,8,13,21);

Flux
  .fromIterable(list)
  ...
```

Create a Flux stream of Integer objects from a Java List collection

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Factory Method Operators in the Flux Class

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava's method Observable. fromIterable() works the same



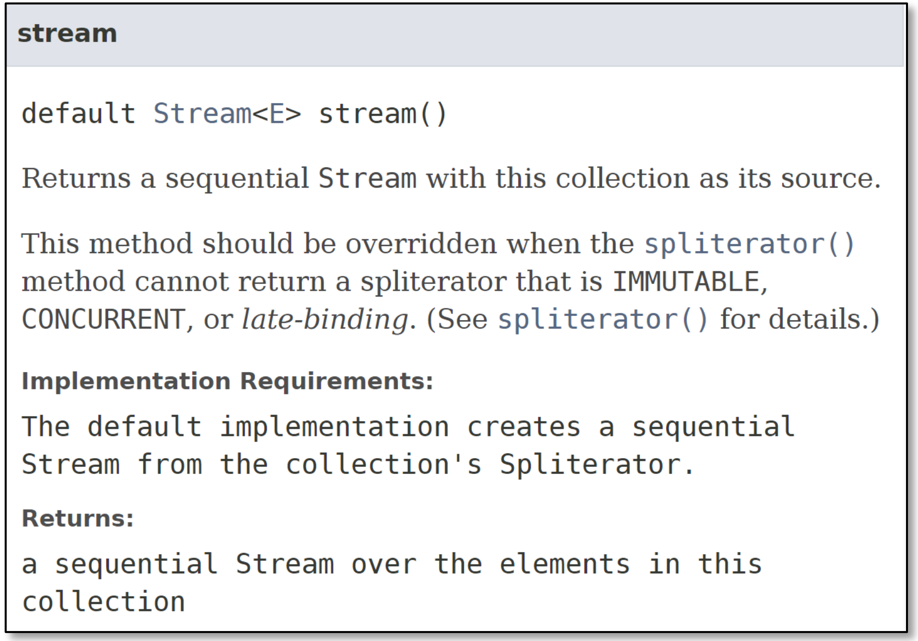
```
List<Integer> list =
  List.of(0,1,1,2,3,5,8,13,21);

Observable
  .fromIterable(list)
  ...
```

Create an Observable stream of Integer objects from a List collection

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromIterable

Key Factory Method Operators in the Flux Class

- The fromIterable() method
  - Create a Flux that emits items contained in the given Iterable
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava's method Observable.fromIterable() works the same
  - Similar to the stream() method in Java Collection



stream

```
default Stream<E> stream()
```

Returns a sequential `Stream` with this collection as its source.

This method should be overridden when the `spliterator()` method cannot return a spliterator that is `IMMUTABLE`, `CONCURRENT`, or *late-binding*. (See `spliterator()` for details.)

Implementation Requirements:

`The default implementation creates a sequential Stream from the collection's Spliterator.`

Returns:

`a sequential Stream over the elements in this collection`

```
List<Integer> list =
  List.of(0,1,1,2,3,5,8,13,21);

list.stream()...
```

Create a stream of Integer objects

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

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array

```
static <T> Flux<T> fromArray
   (T[] array)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#fromArray

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
    - The param provides the array to read the data from

```
static <T> Flux<T> fromArray
   (T[] array)
```

See docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

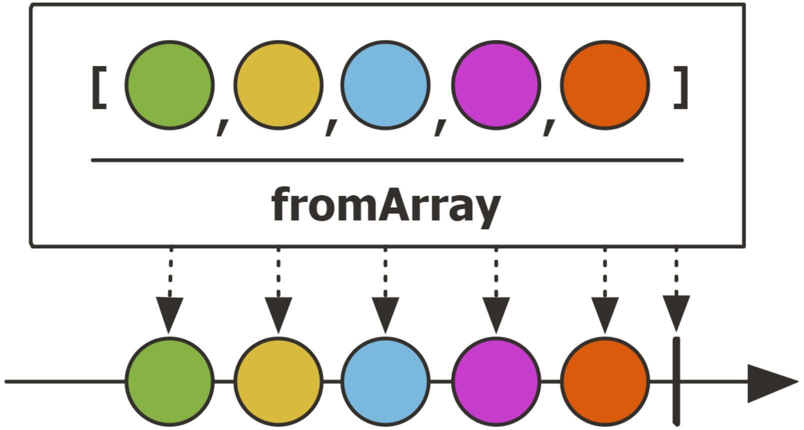
Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
    - The param provides the array to read the data from
    - The returned Flux emits the items from the array

```
static <T> Flux<T> fromArray
   (T[] array)
```

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
  - This factory method operator also adapts non-reactive input sources into the reactive model



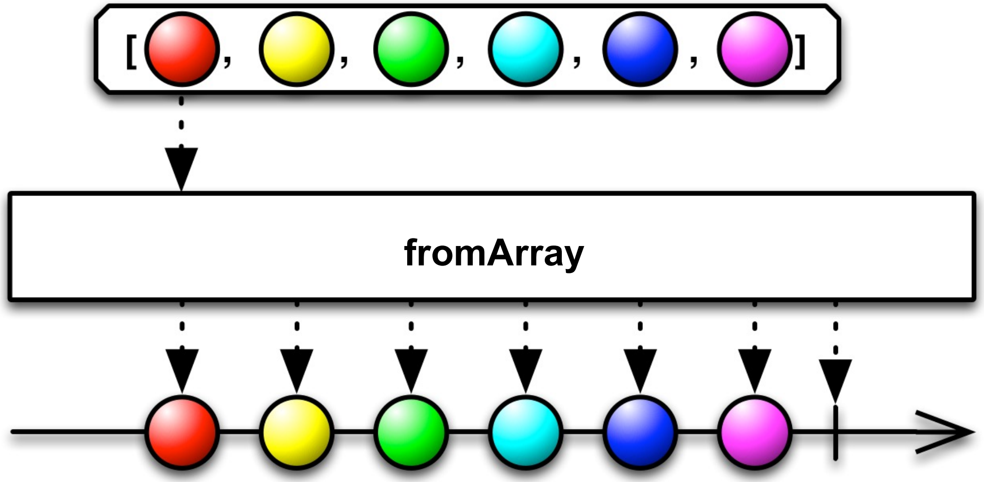
```
Integer[] array =
  {0, 1, 1, 2, 3, 5, 8, 13, 21};

Flux
  .fromArray(array)
  ...
```

Create a Flux stream of Integer objects from a Java built-in array

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava's method Observable.fromArray() works the same



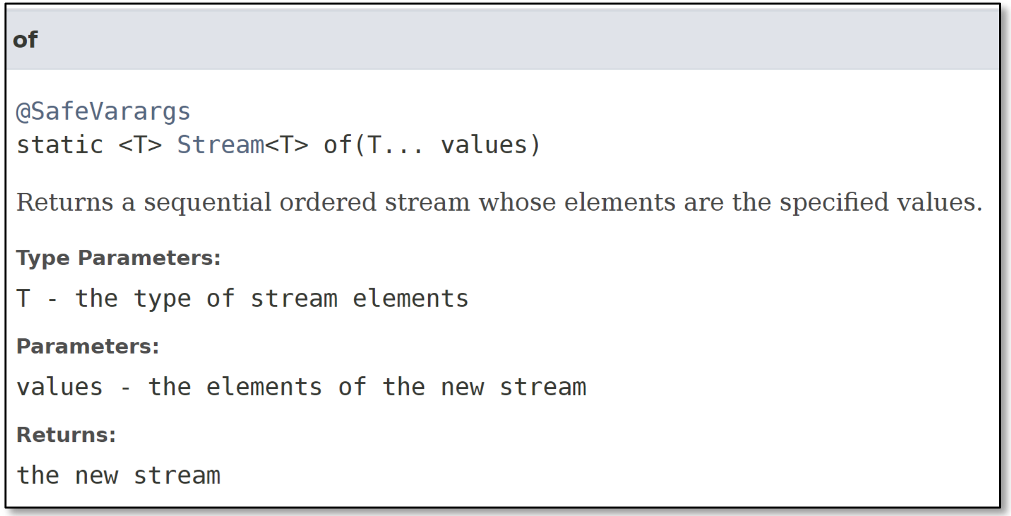
```
Integer[] array =
  {0, 1, 1, 2, 3, 5, 8, 13, 21};

Observable
  .fromArray(array)
  ...
```

Create an Observable stream of Integer objects from a built-in array

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromArray

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava's method Observable. fromArray() works the same
  - Similar to the of() method in Java Streams



of

```
@SafeVarargs
static <T> Stream<T> of(T... values)
```

Returns a sequential ordered stream whose elements are the specified values.

Type Parameters:

`T - the type of stream elements`

Parameters:

`values - the elements of the new stream`

Returns:

`the new stream`

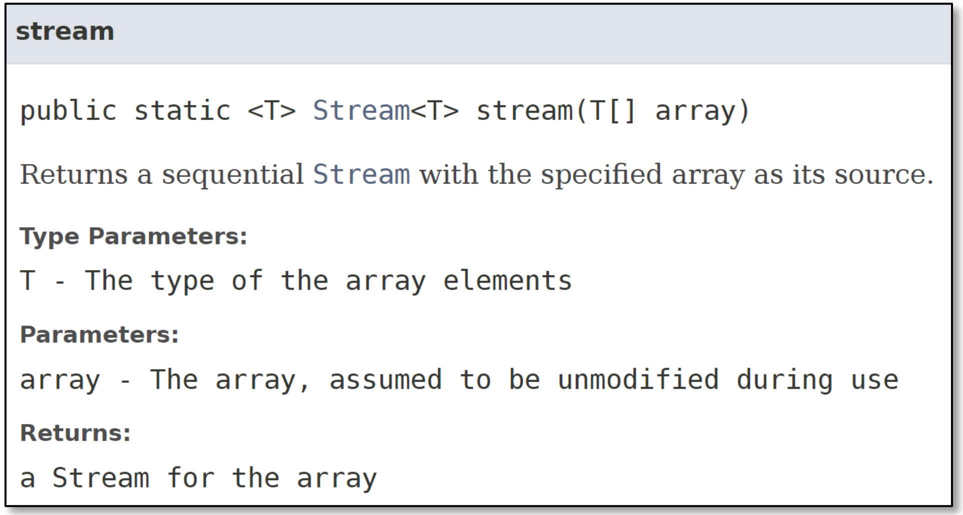
```
Integer[] array =
  {0, 1, 1, 2, 3, 5, 8, 13, 21};

Stream
  .of(array)
  ...
```

Create a stream of Integer objects from a built-in array

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

Key Factory Method Operators in the Flux Class

- The fromArray() method
  - Create a Flux that emits items in the given Java built-in array
  - This factory method operator also adapts non-reactive input sources into the reactive model
  - RxJava's method Observable.fromArray() works the same
  - Similar to the of() method in Java Streams
    - Also, similar to the stream() method in Java Arrays



stream

```
public static <T> Stream<T> stream(T[] array)
```

Returns a sequential Stream with the specified array as its source.

Type Parameters:

T - The type of the array elements

Parameters:

array - The array, assumed to be unmodified during use

Returns:

a Stream for the array

```
Integer[] array =
  {0, 1, 1, 2, 3, 5, 8, 13, 21};

Arrays
  .stream(array)
  ...
```

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

Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API

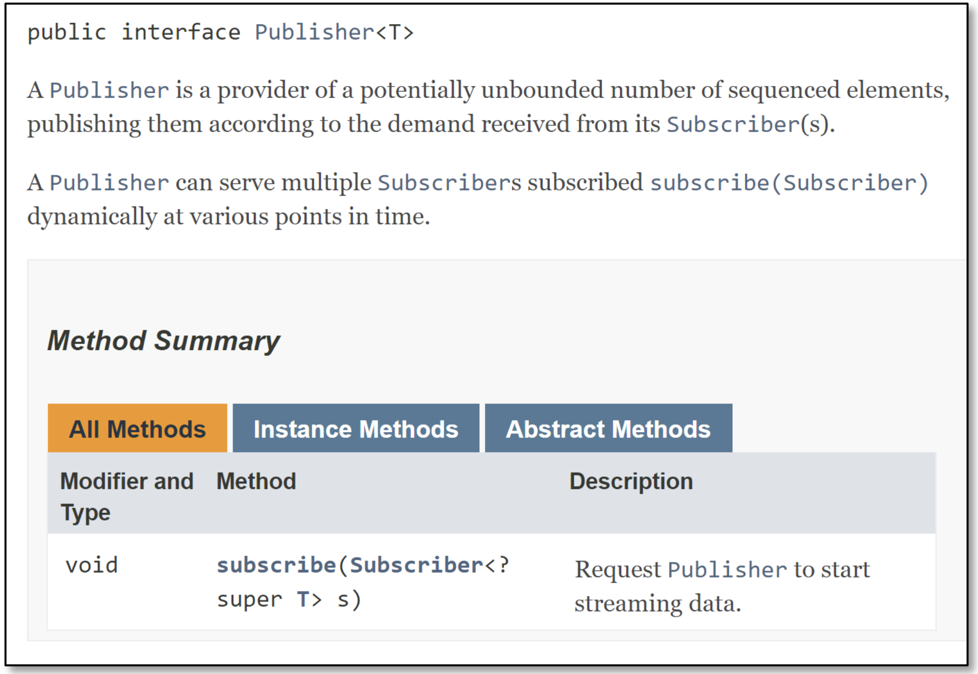
```
static <T> Flux<T> from
  (Publisher<? extends T> source)
```

See projectreactor.io/docs/core/release/api/reactor/core/publisher/Flux.html#from

Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
    - The param provides the source to decorate

```
static <T> Flux<T> from
   (Publisher<? extends T> source)
```



`public interface Publisher<T>`

A `Publisher` is a provider of a potentially unbounded number of sequenced elements, publishing them according to the demand received from its `Subscriber`(s).

A `Publisher` can serve multiple `Subscribers` subscribed `subscribe(Subscriber)` dynamically at various points in time.

Method Summary

All Methods | **Instance Methods** | **Abstract Methods**

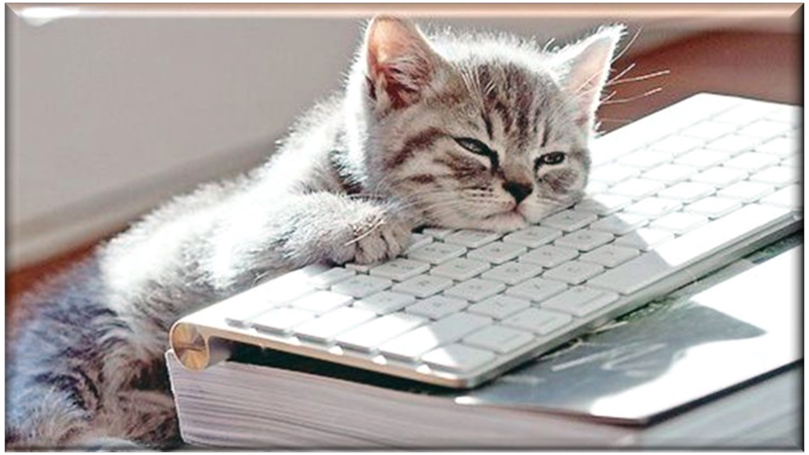
Modifier and Type	Method	Description
`void`	`subscribe(Subscriber<? super T> s)`	Request `Publisher` to start streaming data.

See www.reactive-streams.org/reactive-streams-1.0.3-javadoc/org/reactivestreams/Publisher.html

Key Factory Method Operators in the Flux Class

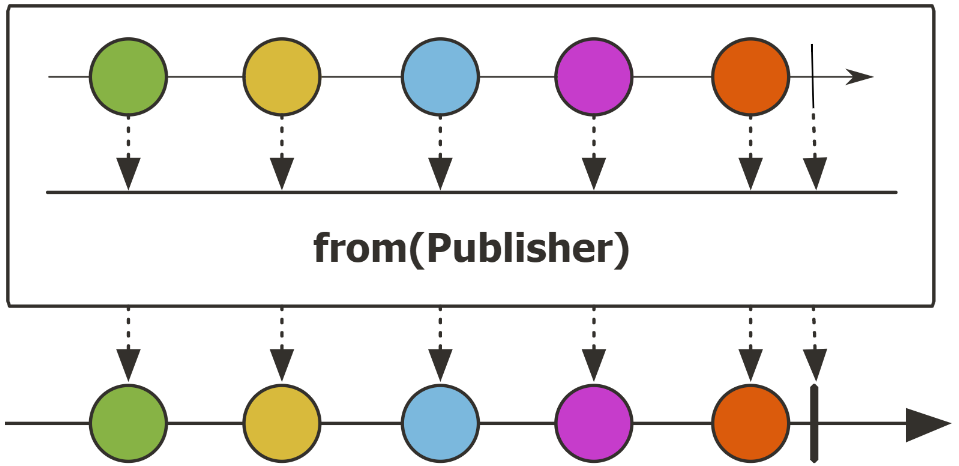
- The from() method
  - Decorate the specified Publisher with the Flux API
    - The param provides the source to decorate
    - Returns a new Flux that decorates the source at runtime
      - i.e., it's "lazy"

```
static <T> Flux<T> from
   (Publisher<? extends T> source)
```



Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
  - This factory method operator adapts non-Flux publishers into the Flux API



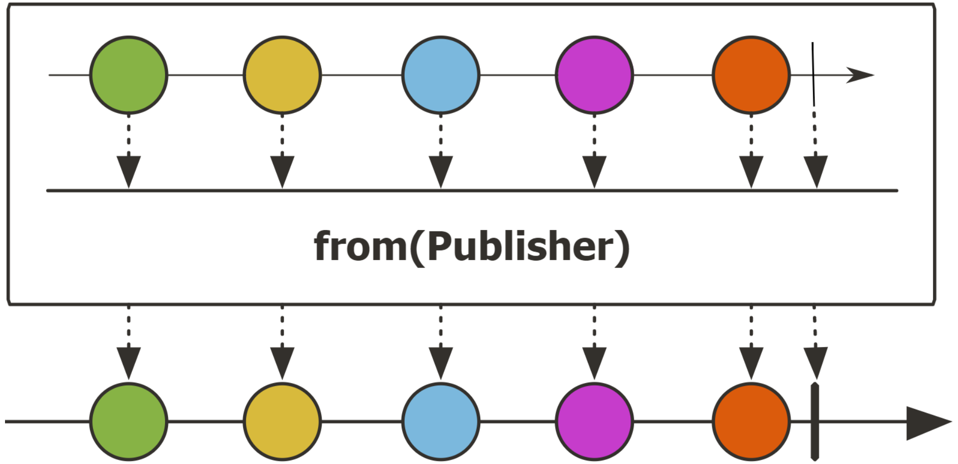
```
Flux
   .from(Mono
           .fromCallable
              (() ->
               BigFractionUtils
                .makeBigFraction(random,
                                 true)))
```

Create a Flux containing a single BigFraction object from a Mono

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
  - This factory method operator adapts non-Flux publishers into the Flux API
    - from() is "lazy"

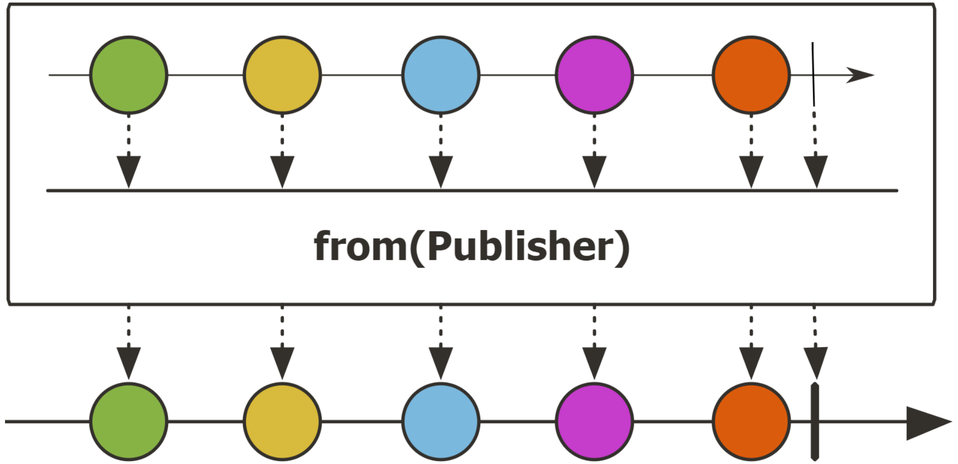


```
Flux
  .from(Mono
          .fromCallable
             (() ->
              BigFractionUtils
               .makeBigFraction(random,
                                true)))
```

It invokes the Publisher param at the time of subscription & separately for each subscriber

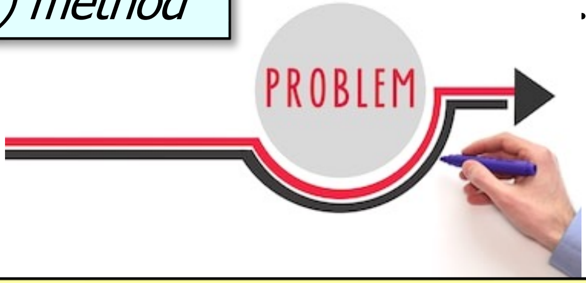
Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
  - This factory method operator adapts non-Flux publishers into the Flux API
    - from() is "lazy"



Can be used to workaround Flux's lack of a fromCallable() method

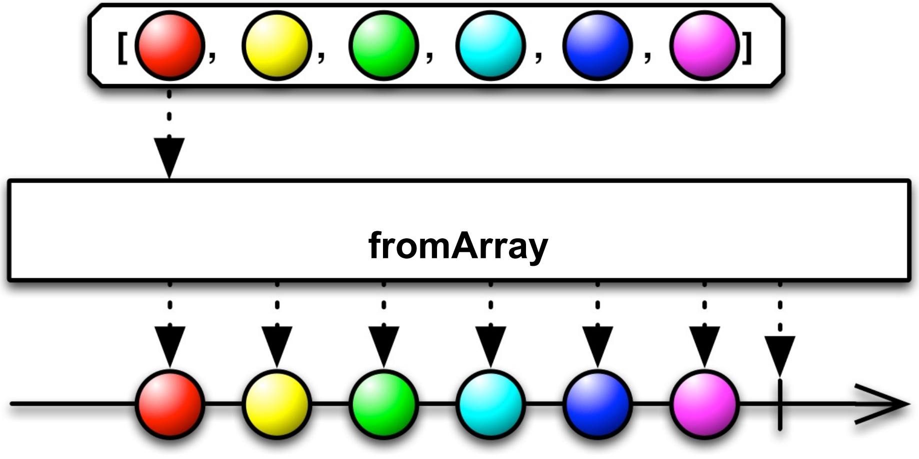
```
Flux
 .from(Mono
         .fromCallable
            (() ->
             BigFractionUtils
              .makeBigFraction(random,
                                true)))
```



See chat.openai.com/share/17ba266c-39f4-4834-84bf-dd8254a65be3

Key Factory Method Operators in the Flux Class

- The from() method
  - Decorate the specified Publisher with the Flux API
  - This factory method operator adapts non-Flux publishers into the Flux API
  - RxJava's method Observable. fromCallable() is similar



```
Observable
  .fromCallable
    (() ->
     BigFractionUtils
       .makeBigFraction(random,
                        true)))
```

Create an Observable containing a single BigFraction object

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Observable.html#fromCallable

End of Key Factory Method Operators in the Flux Class (Part 1)