Key Combining 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
  - Factory method operators
  - Transforming operators
  - Action operators
  - Combining operators
    - These operators create a Flux from multiple iterations or sources
      - e.g., repeat() & mergeWith()

Key Combining Operators in the Flux Class

Key Combining Operators in the Flux Class

- The repeat() operator
  - Returns a Flux that repeats the sequence of items emitted by the given Flux numRepeat # of times

```
Flux<T> repeat(long numRepeat)
```

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

Key Combining Operators in the Flux Class

- The repeat() operator
  - Returns a Flux that repeats the sequence of items emitted by the given Flux numRepeat # of times
    - This results in numRepeat + 1 total subscriptions to the original source
      - As a consequence, using 0 plays the original sequence once

```
Flux<T> repeat(long numRepeat)
```

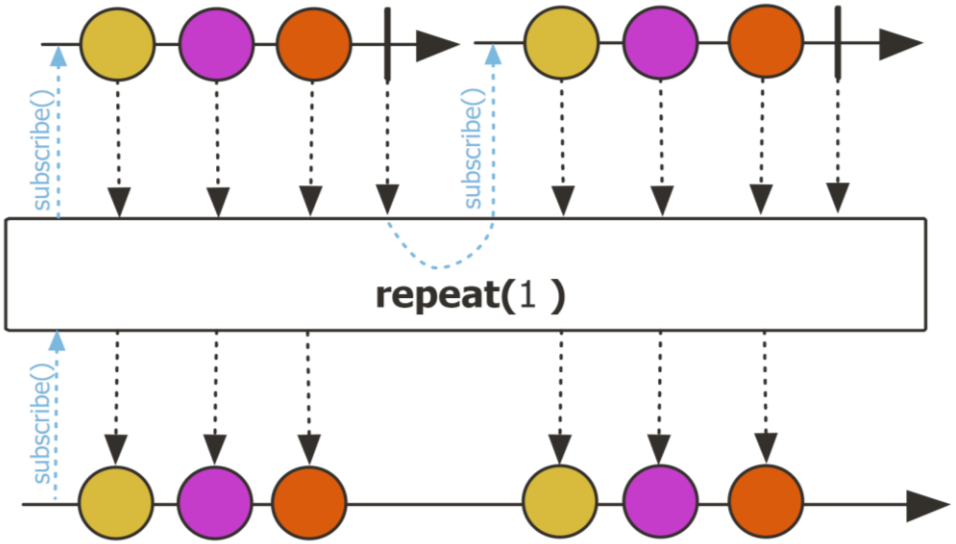
Key Combining Operators in the Flux Class

- The repeat() operator
  - Returns a Flux that repeats the sequence of items emitted by the given Flux numRepeat # of times
    - This results in numRepeat + 1 total subscriptions to the original source
  - Returns a new Flux instance

```
Flux<T> repeat(long numRepeat)
```

Key Combining Operators in the Flux Class

- The repeat() operator
  - Returns a Flux that repeats the sequence of items emitted by the given Flux numRepeat # of times
  - This method does not operate by default on a particular Scheduler



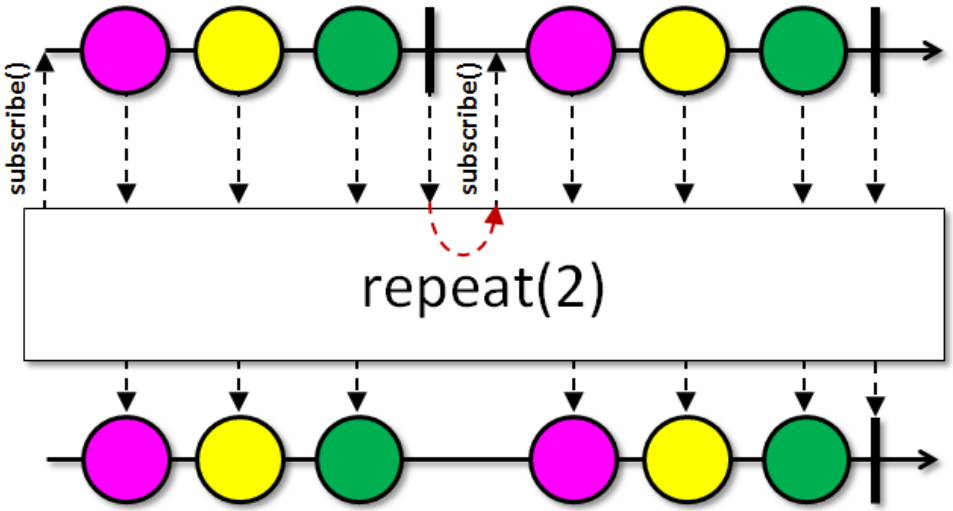

```
Flux.from
    (Mono.fromCallable(() ->
        BigFractionUtils.
            makeBigFraction
              (random, true)))
.repeat(3);
```

Generate 4 random, reduced big fractions

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

Key Combining Operators in the Flux Class

- The repeat() operator
  - Returns a Flux that repeats the sequence of items emitted by the given Flux numRepeat # of times
  - This method does not operate by default on a particular Scheduler
  - RxJava's Observable.repeat() works the same




```
Observable
    .fromCallable(() ->
        BigFractionUtils.
            makeBigFraction
              (random, true))
    .repeat(3);
```

Generate 4 random, reduced big fractions

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

Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence

```
Flux<T> mergeWith
   (Publisher<? extends T> other)
```

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

Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
    - The param is the Publisher to merge with

```
Flux<T> mergeWith
  (Publisher<? extends T> other)
```

Interface Publisher<T>

Type Parameters:

T - the type of element signaled.

All Known Subinterfaces:

Processor<T,R>

```
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.

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

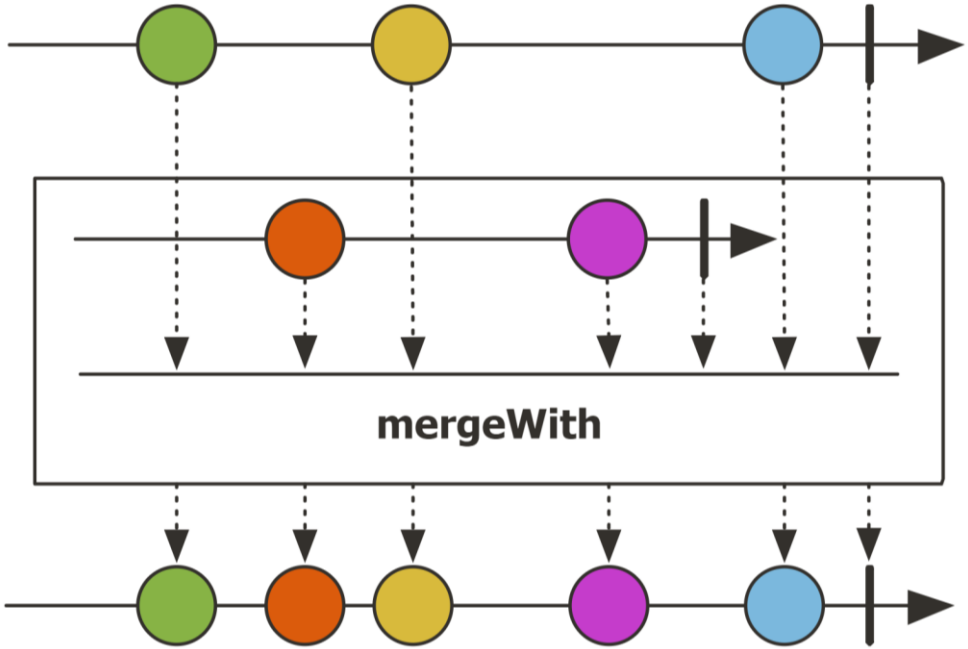
Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
    - The param is the Publisher to merge with
    - Returns the new merged Flux instance

```
Flux<T> mergeWith
   (Publisher<? extends T> other)
```

Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
  - This method combines items emitted by multiple Flux sources so that they appear as a single Flux

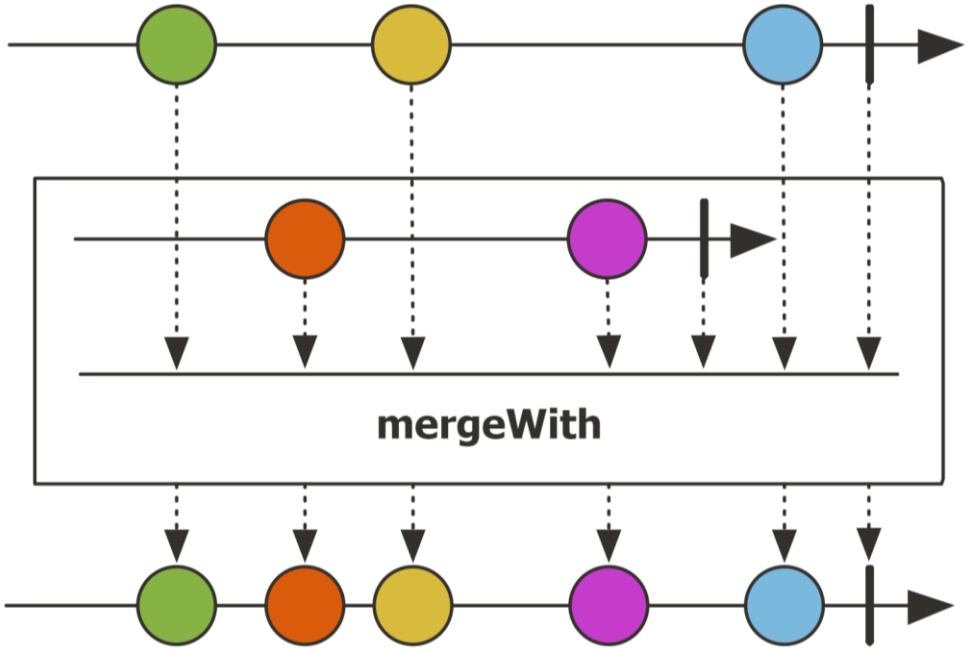



```
Flux<BigFraction> f1 ...
Flux<BigFraction> f2 ...
f1.mergeWith(f2)...
```

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

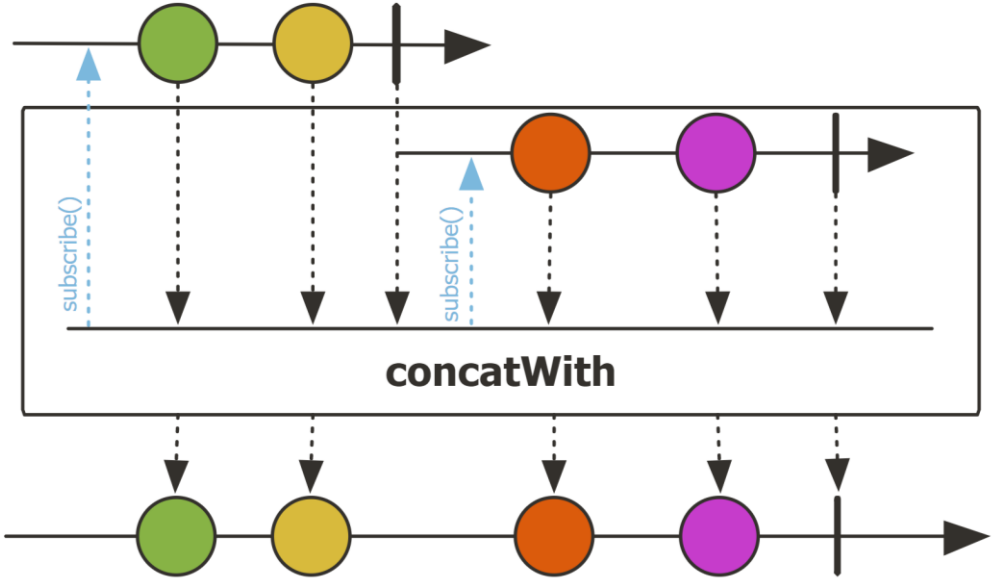
Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
  - This method combines items emitted by multiple Flux sources so that they appear as a single Flux
    - This merging may interleave the items



Key Combining Operators in the Flux Class

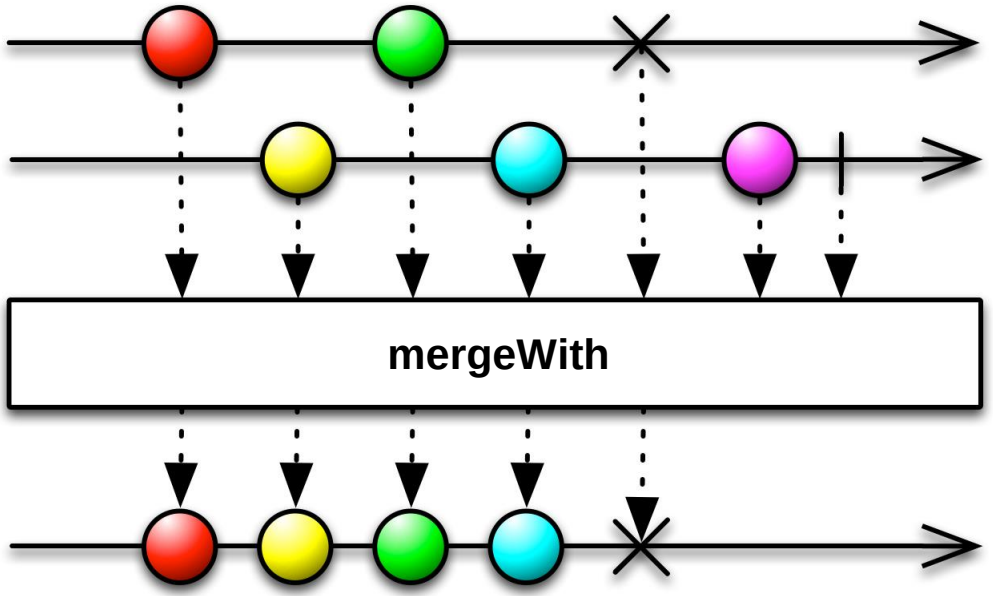
- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
  - This method combines items emitted by multiple Flux sources so that they appear as a single Flux
    - This merging may interleave the items
    - Use concatWith() to avoid interleaving



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

Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
  - This method combines items emitted by multiple Flux sources so that they appear as a single Flux
  - RxJava's method Observable.mergeWith() works the same




```
Observable<BigFraction> o1 ...
Observable<BigFraction> o2 ...
o1.mergeWith(o2)...
```

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

Key Combining Operators in the Flux Class

- The mergeWith() operator
  - Merge data from this Flux & a Publisher into an interleaved merged sequence
  - This method combines items emitted by multiple Flux sources so that they appear as a single Flux
  - RxJava's method Observable.mergeWith() works the same
- Similar to the Stream.concat() method in Java Streams

concat

```
static <T> Stream<T> concat(Stream<? extends T> a,
                            Stream<? extends T> b)
```

Creates a lazily concatenated stream whose elements are all the elements of the first stream followed by all the elements of the second stream. The resulting stream is ordered if both of the input streams are ordered, and parallel if either of the input streams is parallel. When the resulting stream is closed, the close handlers for both input streams are invoked.

```
List<String> concats
          (List<String> l, int n) {
  Stream<String> s = Stream.empty();
  while (--n >= 0)
    s = Stream.concat(s, l.stream());
  return s.collect(toList());
}
```

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

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