Key Combining 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 operations
  - Transforming operators
  - Concurrency & scheduler operators
  - Error handling operators
  - Combining operators
    - These operators create a Flux from multiple sources or iterations
      - e.g., reduce(), collectList(), & collect()

Key Combining Operators in the Flux Class

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items

```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```

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

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param

```
Mono<U> reduce
    (BiFunction<T, T, T> reducer)
```

Interface BiFunction<T,U,R>

Type Parameters:

T - the type of the first argument to the function

U - the type of the second argument to the function

R - the type of the result of the function

All Known Subinterfaces:

BinaryOperator<T>

Functional Interface:

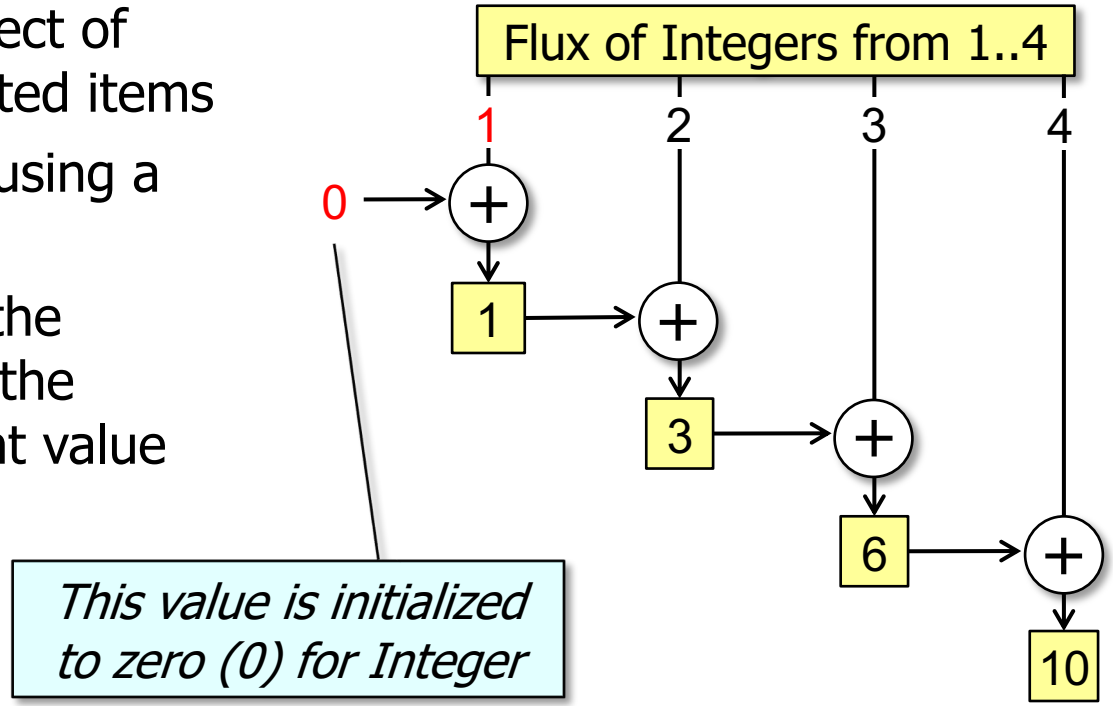
This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

See docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value

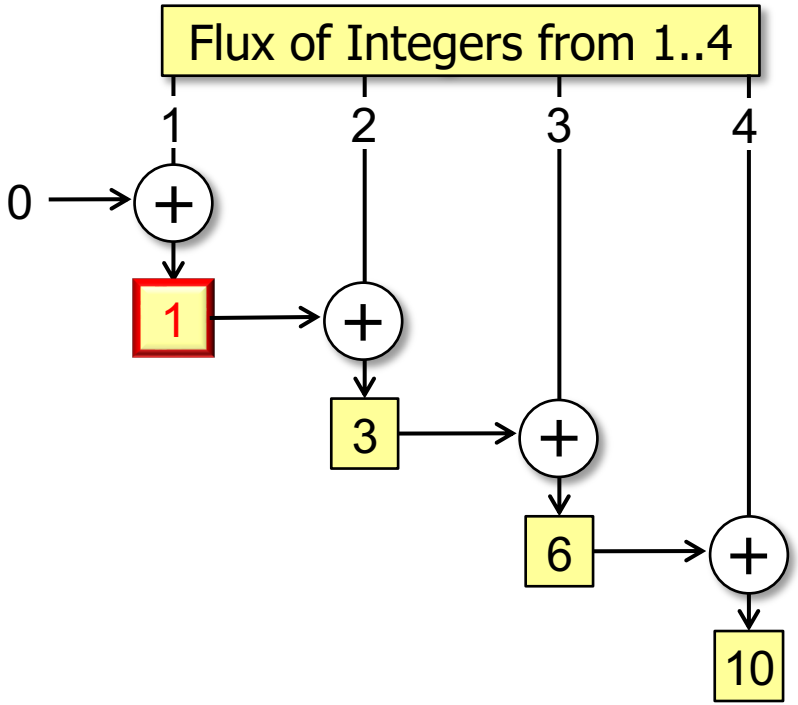
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction

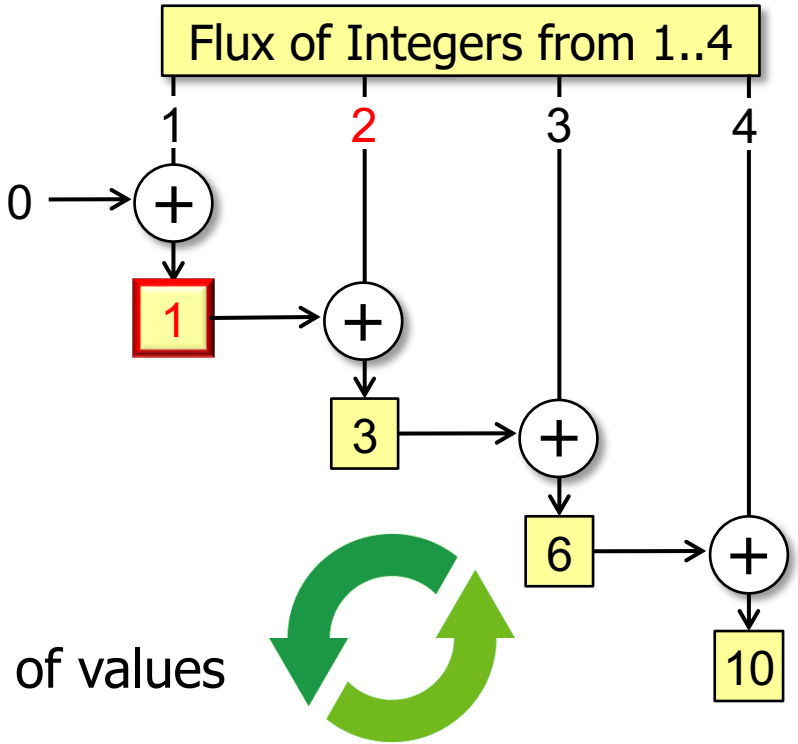
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

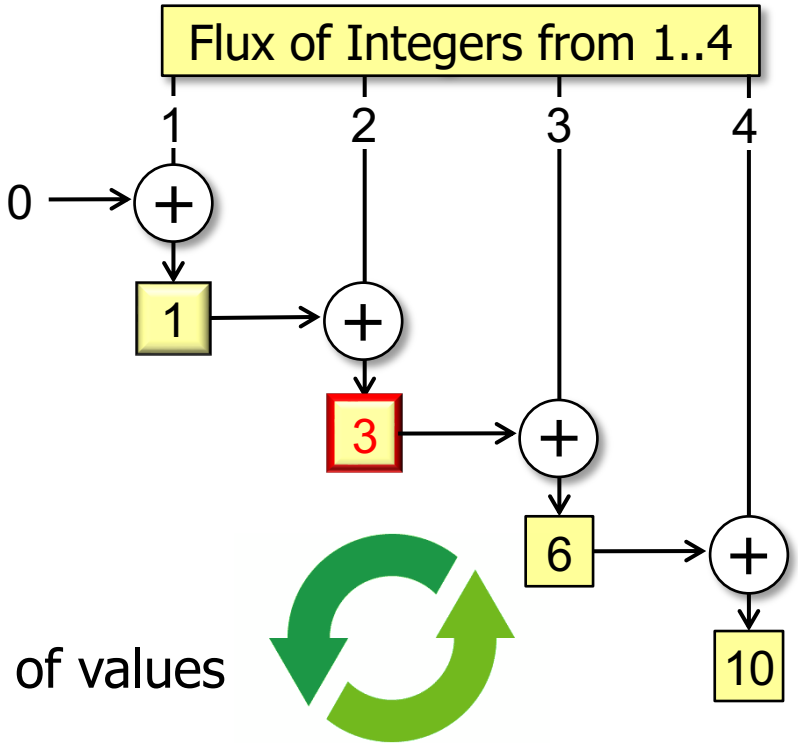
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

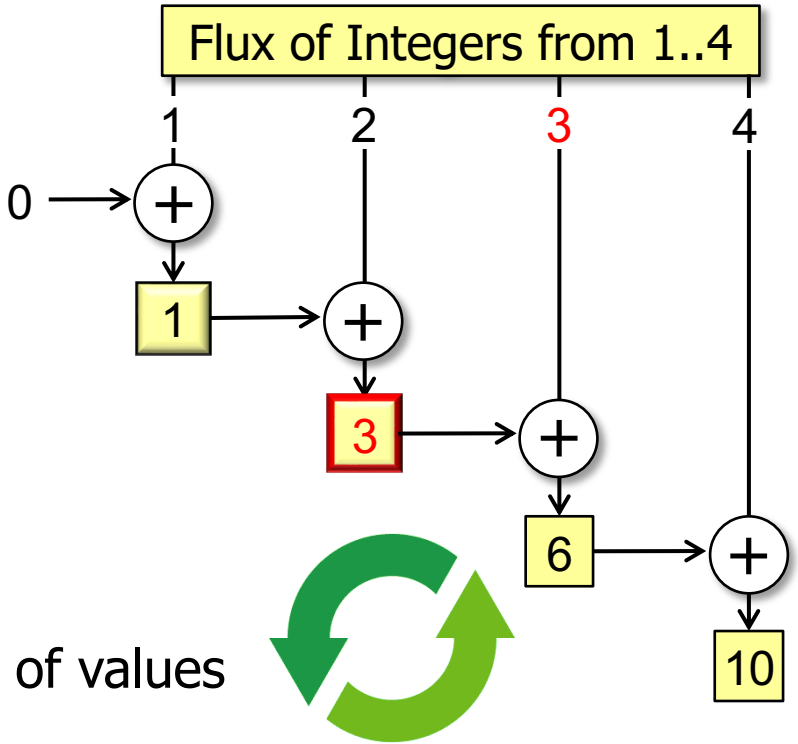
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

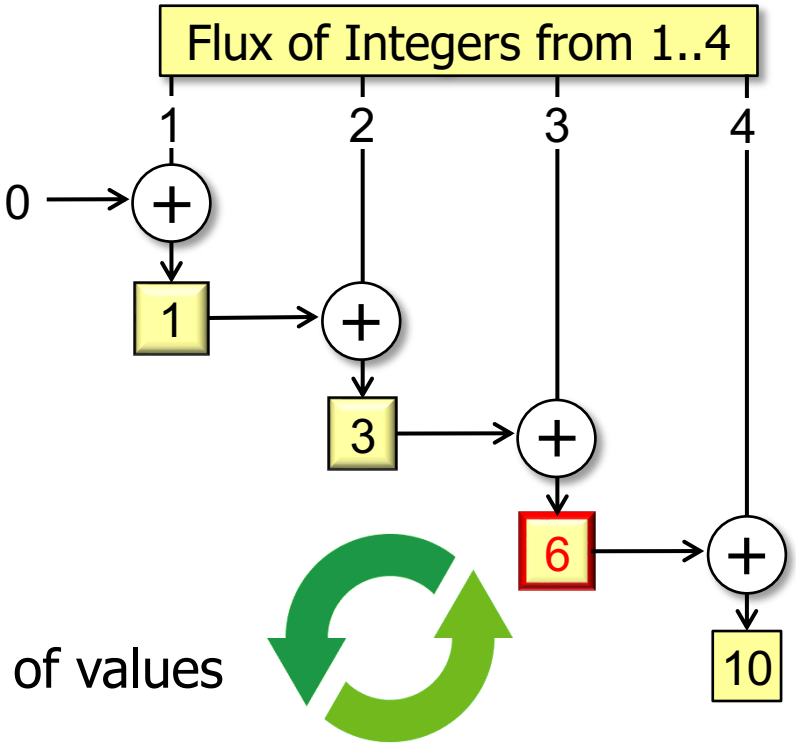
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

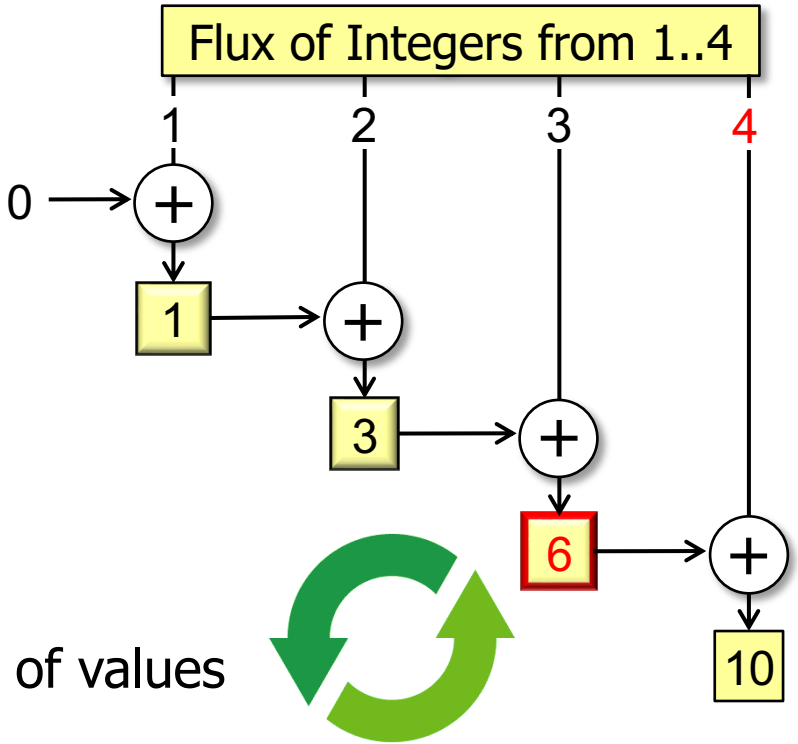
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

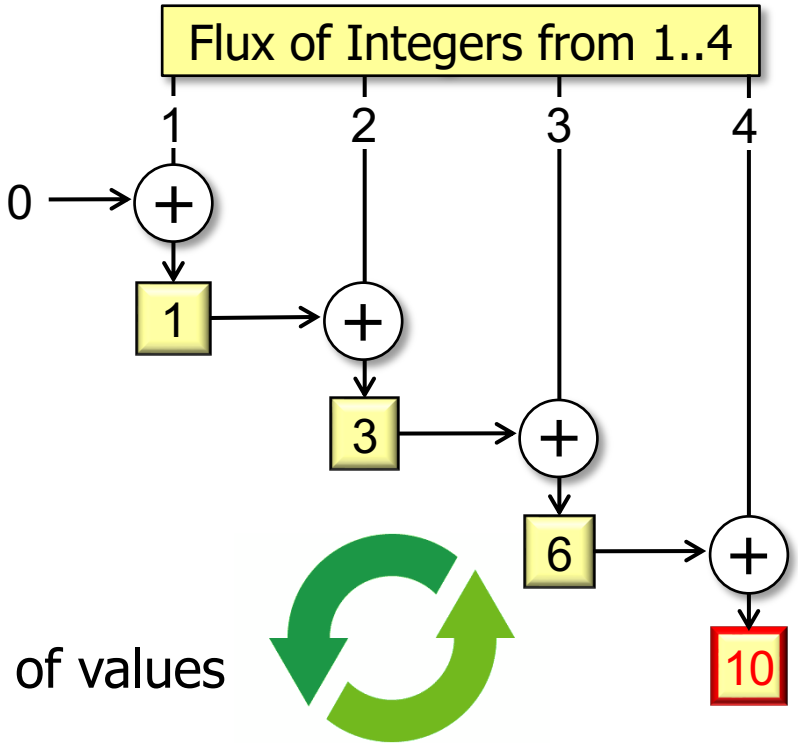
```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
      - This param is passed the intermediate result of the reduction & the current value
        - It returns the next intermediate value of the reduction
      - The process repeats for each pair of values

```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The result of the reduced Flux is emitted from the final call as sole item of a Mono

```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The result of the reduced Flux is emitted from the final call as sole item of a Mono
    - If the Flux emits no items Mono will be empty

```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```

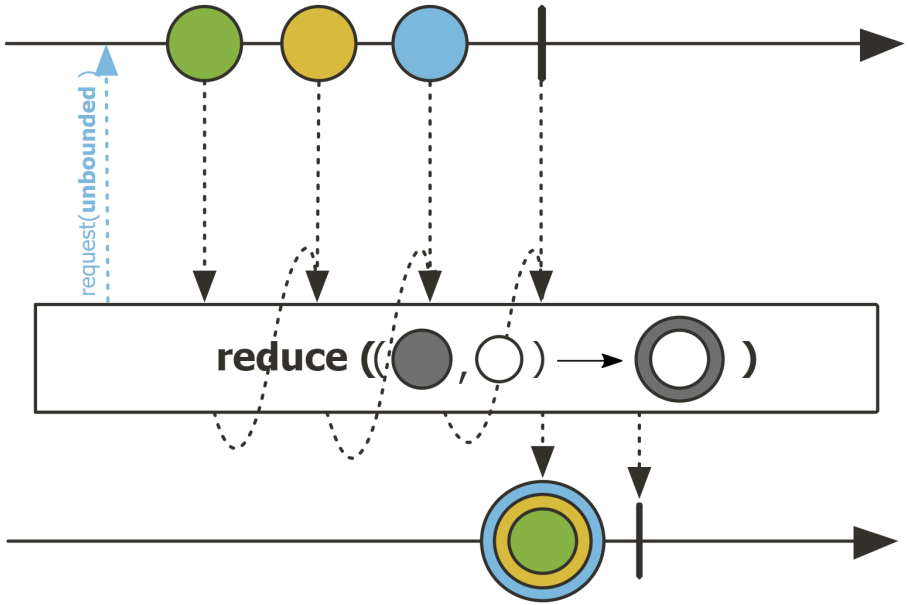
Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The result of the reduced Flux is emitted from the final call as sole item of a Mono
    - If the Flux emits no items Mono will be empty
    - The internally accumulated value is discarded upon cancellation or error

```
Mono<U> reduce
   (BiFunction<T, T, T> reducer)
```

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted



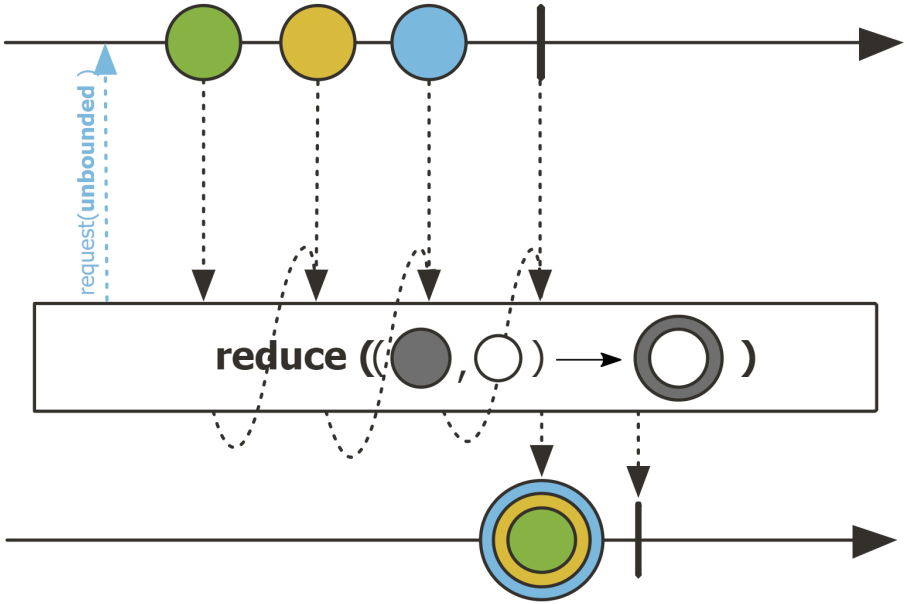

```
return Flux
   .fromArray(bigFractions)
   .flatMap(bf ->
            multiplyFractions(bf,
                              Schedulers.parallel()))
   .reduce(BigFraction::add)
   ...
```

Sum results of async multiplications

See Reactive/flux/ex3/src/main/java/FluxEx.java

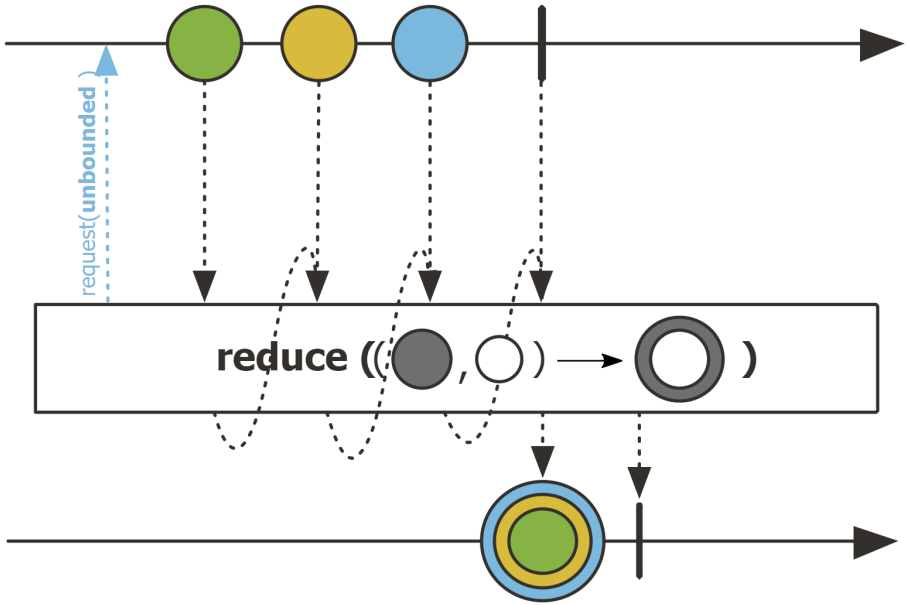
Key Combining Operators in the Flux Class

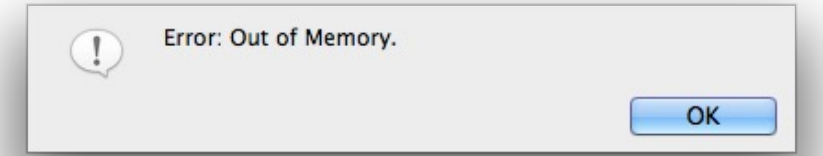
- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
    - Sources that are infinite & never complete will never emit anything through this operator



Key Combining Operators in the Flux Class

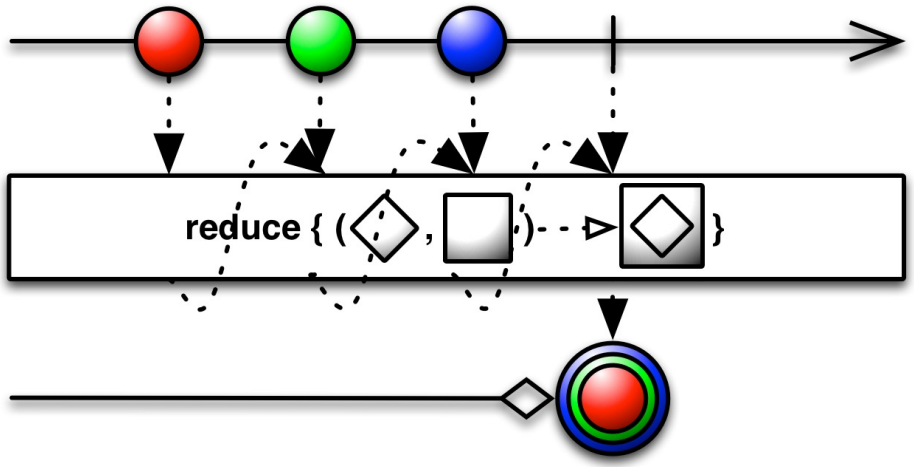
- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
    - Sources that are infinite & never complete will never emit anything through this operator
      - An infinite source may lead to a fatal OutOfMemoryError





Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted



- RxJava's Observable.reduce() operator works the same

```
return Observable
   .fromArray(bigFractions)
   .flatMap(bf ->
            multiplyFrations(bf, Schedulers.computation()))
   .reduce(BigFraction::add) ...
```

Sum the results of async multiplications

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

Key Combining Operators in the Flux Class

- The reduce() operator
  - Reduce the values from this Flux sequence into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
  - RxJava's Observable.reduce() operator works the same
  - Similar to the Stream.reduce() method in Java Streams

```
int result = List
   .of(1, 2, 3, 4, 5, 6).stream()
   .reduce(0, Math::addExact);
```

Sum the #'s together

reduce

`Optional<T> reduce(BinaryOperator<T> accumulator)`

Performs a reduction on the elements of this stream, using an associative accumulation function, and returns an `Optional` describing the reduced value, if any. This is equivalent to:

```
boolean foundAny = false;
T result = null;
for (T element : this stream) {
    if (!foundAny) {
        foundAny = true;
        result = element;
    }
    else
        result = accumulator.apply(result, element);
}
return foundAny ? Optional.of(result) : Optional.empty();
```

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

Key Combining Operators in the Flux Class

- The collectList() operator
  - Collect all elements emitted by this Flux into a List

```
Mono<List<T>> collectList()
```

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

Key Combining Operators in the Flux Class

- The collectList() operator
  - Collect all elements emitted by this Flux into a List
    - Returns a Mono to a List containing all values from this Flux

```
Mono<List<T>> collectList()
```

Class Mono<T>

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

Type Parameters:

T - the type of the single value of this class

All Implemented Interfaces:

Publisher<T>, CorePublisher<T>

Direct Known Subclasses:

MonoOperator, MonoProcessor

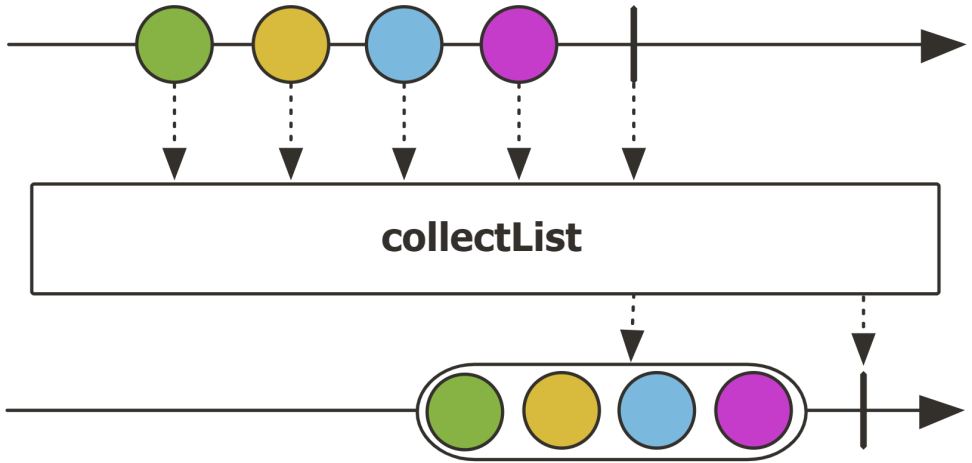
```
public abstract class Mono<T>
extends Object
implements CorePublisher<T>
```

A Reactive Streams `Publisher` with basic rx operators that completes successfully by emitting an element, or with an error.

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

Key Combining Operators in the Flux Class

- The collectList() operator
  - Collect all elements emitted by this Flux into a List
  - The list is emitted by the Mono when this sequence completes



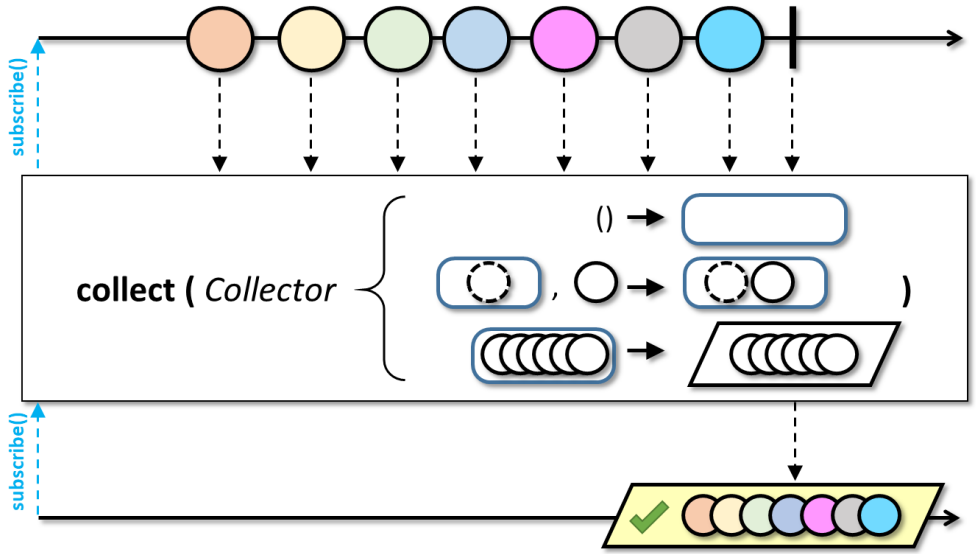

```
Flux
   .fromIterable
      (bigFractions)
   .flatMap(...)
   .filter(fraction -> fraction.compareTo(0) > 0)
   .collectList()
   ...
```

Collect the filtered BigFractions into a list

See Reactive/flux/ex3/src/main/java/FluxEx.java

Key Combining Operators in the Flux Class

- The collectList() operator
  - Collect all elements emitted by this Flux into a List
  - The list is emitted by the Mono when this sequence completes
  - RxJava's Observable.collect() is a generalization of collectList()




```
Observable
   .fromIterable(bigFractions)
   .flatMap(...)
   .filter(fraction -> fraction.compareTo(0) > 0)
   .collect(toList())
   ...
```

Collect the filtered BigFractions into a list

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

Key Combining Operators in the Flux Class

- The collectList() operator
  - Collect all elements emitted by this Flux into a List
  - The list is emitted by the Mono when this sequence completes
  - RxJava's Observable.collect() is a generalization of collectList()
- Similar to the Stream.toList() method in Java Streams

collect

`<R,A> R collect(Collector<? super T,A,R> collector)`

Performs a mutable reduction operation on the elements of this stream using a `Collector`. A `Collector` encapsulates the functions used as arguments to `collect(Supplier, BiConsumer, BiConsumer)`, allowing for reuse of collection strategies and composition of collect operations such as multiple-level grouping or partitioning.

```
List<Integer> evenNumbers = List
   .of(1, 2, 2, 3, 4, 5, 6, 6)
   .stream()
   .filter(x -> x % 2 == 0)
   .toList();
```

Collect even #'d Integers into a List

See docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/stream/Stream.html#toList()

Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container

```
<R, A> Mono<R> collect
   (Collector<? super T,
                A,
                ? extends R> collector)
```

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

Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container
    - The param is the Java Stream Collector interface
      - This interface defines the supplier(), accumulator(), combiner(), & finisher() methods

```
<R, A> Mono<R> collect
   (Collector<? super T,
              A,
              ? extends R> collector)
```

Interface Collector<T,A,R>

Type Parameters:

T - the type of input elements to the reduction operation

A - the mutable accumulation type of the reduction operation (often hidden as an implementation detail)

R - the result type of the reduction operation

---

public interface **Collector<T,A,R>**

A mutable reduction operation that accumulates input elements into a mutable result container, optionally transforming the accumulated result into a final representation after all input elements have been processed. Reduction operations can be performed either sequentially or in parallel.

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

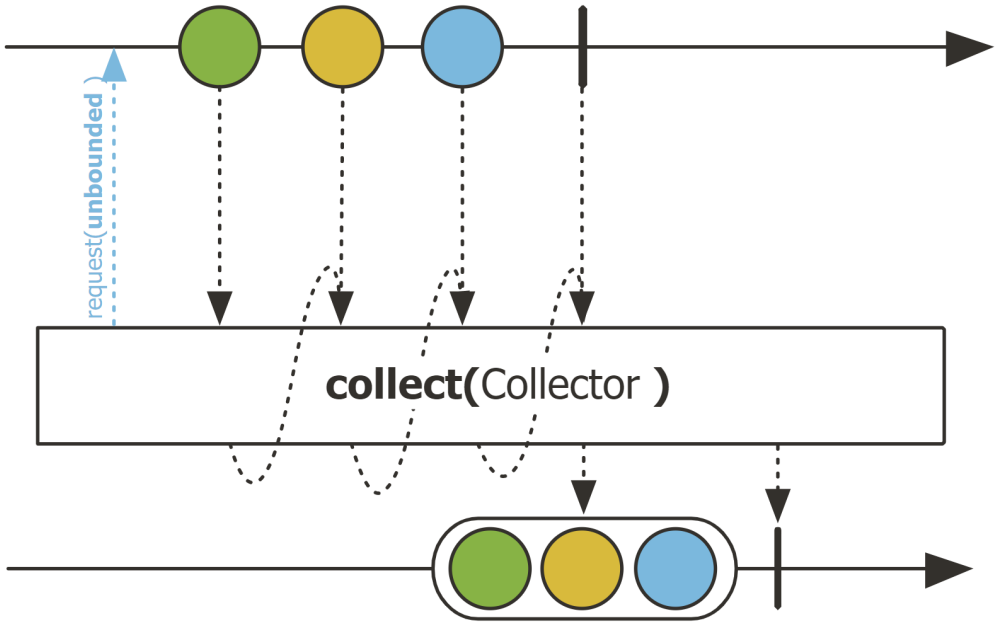
Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container
    - The param is the Java Stream Collector interface
  - The collected result is emitted via a Mono when this sequence completes

```
<R, A> Mono<R> collect
  (Collector<? super T,
             A,
             ? extends R> collector)
```

Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container
  - Can be used to seamlessly integrate Project Reactor & Java Streams capabilities



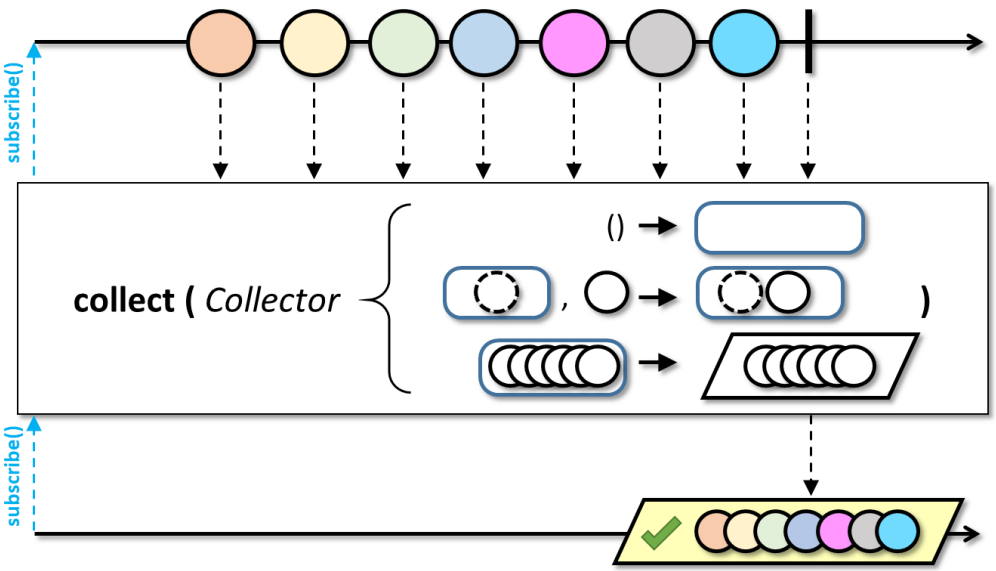

```
return monos -> Mono
    .when(monos)
    .materialize()
    .flatMap(v -> Flux
                .fromIterable(monos)
                .map(Mono::block)
                .collect(toList()));
```

Return a Mono to a List of results that were computed asynchronously

See Reactive/flux/ex3/src/main/java/utils/MonosCollector.java

Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container
  - Can be used to seamlessly integrate Project Reactor & Java Streams capabilities
  - RxJava's operator Observable.collect() works the same




```
Observable
   .fromIterable(bigFractions)
   .flatMap(...)
   .filter(fraction -> fraction.compareTo(0) > 0)
   .collect(toList())
   ...
```

Collect the filtered BigFractions into a list

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

Key Combining Operators in the Flux Class

- The collect() operator
  - Collect all elements emitted by this Flux into a container
  - Can be used to seamlessly integrate Project Reactor & Java Streams capabilities
  - RxJava's operator Observable.collect() works the same
  - Similar to the Stream.collect() method in Java Streams

collect

```
<R,A> R collect(Collector<? super T,A,R> collector)
```

Performs a mutable reduction operation on the elements of this stream using a `Collector`. A `Collector` encapsulates the functions used as arguments to `collect(Supplier, BiConsumer, BiConsumer)`, allowing for reuse of collection strategies and composition of collect operations such as multiple-level grouping or partitioning.

```
Set<Integer> evenNumbers = List
   .of(1, 2, 2, 3, 4, 4, 5, 6, 6)
   .stream()
   .filter(x -> x % 2 == 0)
   .collect(toSet());
```

Collect even #'d Integers into a Set of unique values

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

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