Key Combining Operators in the Observable 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 Observable operators
  - Factory method operations
  - Transforming operators
  - Concurrency & scheduler operators
  - Error handling operators
  - Combining operators
    - This operator creates a Maybe by accumulating elements in an Observable stream
      - e.g., reduce()

Key Combining Operators in the Observable Class

Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values into a single object of the same type as the emitted items

```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```

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

Key Combining Operators in the Observable Class

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

```
Maybe<T> 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:

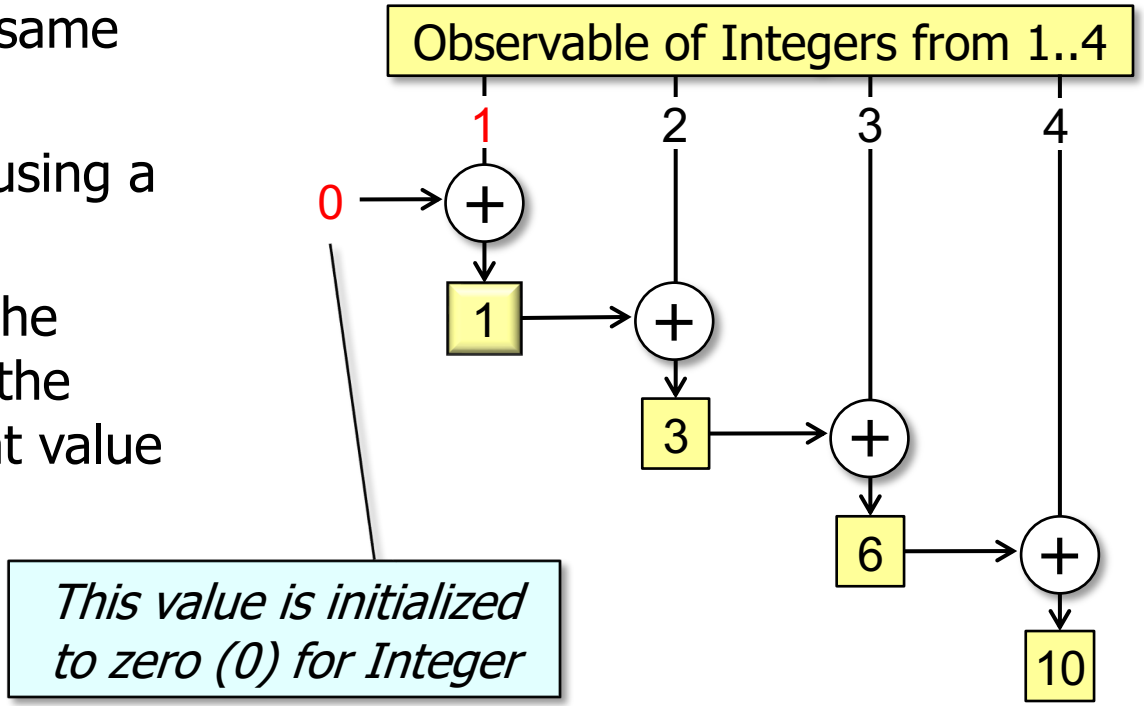
This is a functional interface and can therefore be used as the assignment target for a lambda expression or method reference.

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/functions/BiFunction.html

Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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

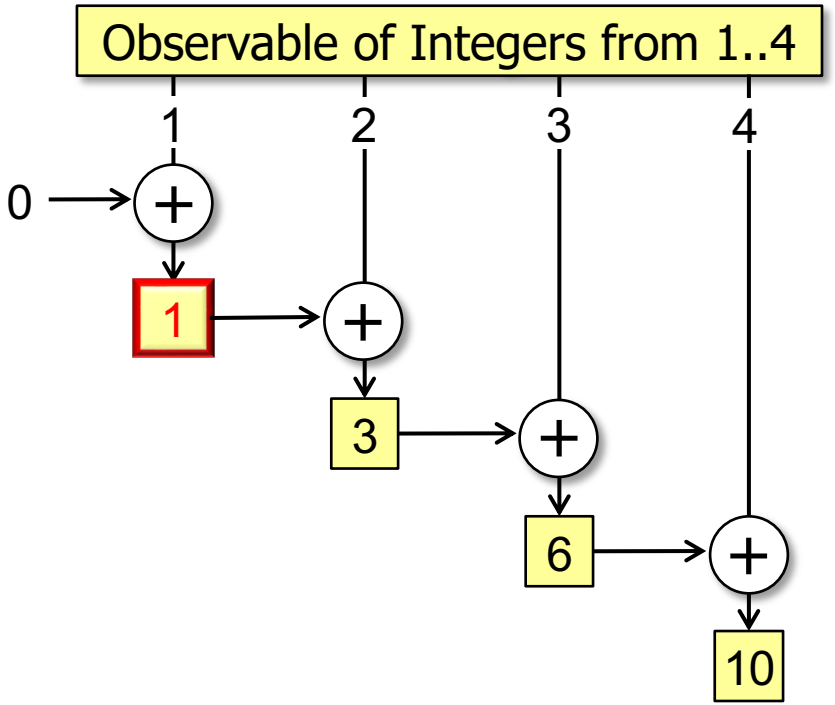
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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

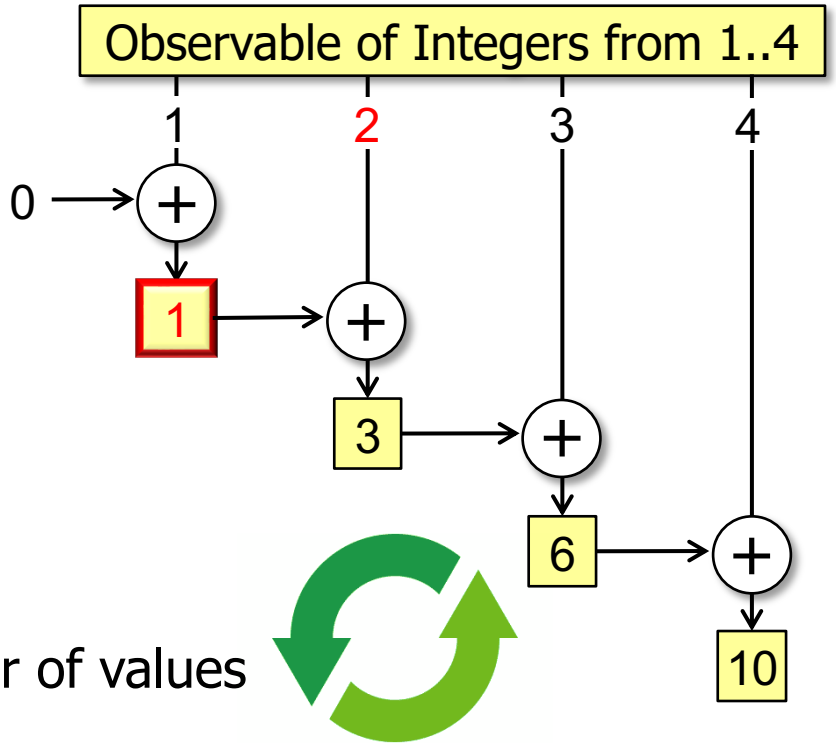
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

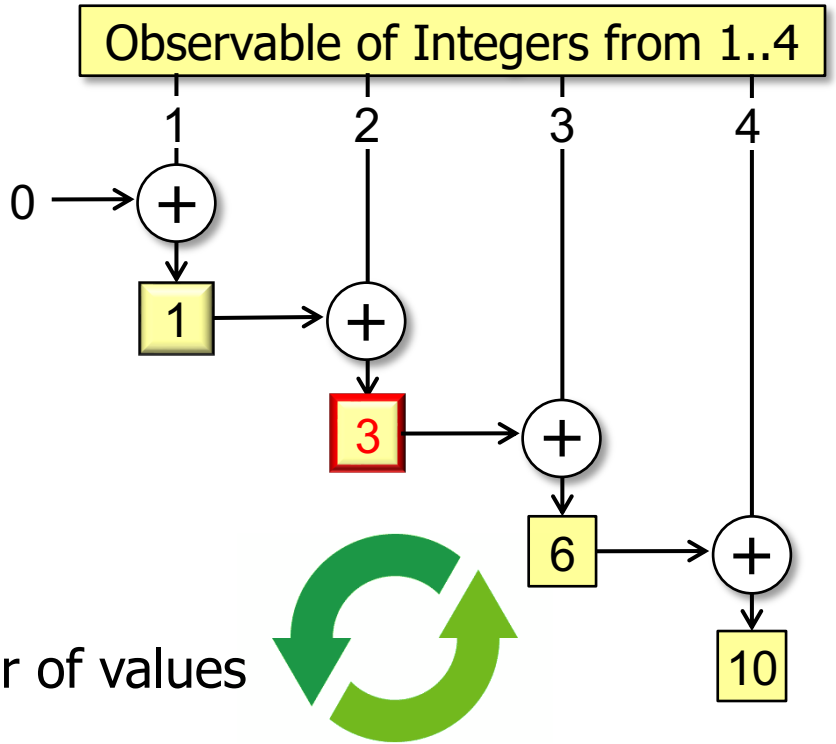
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

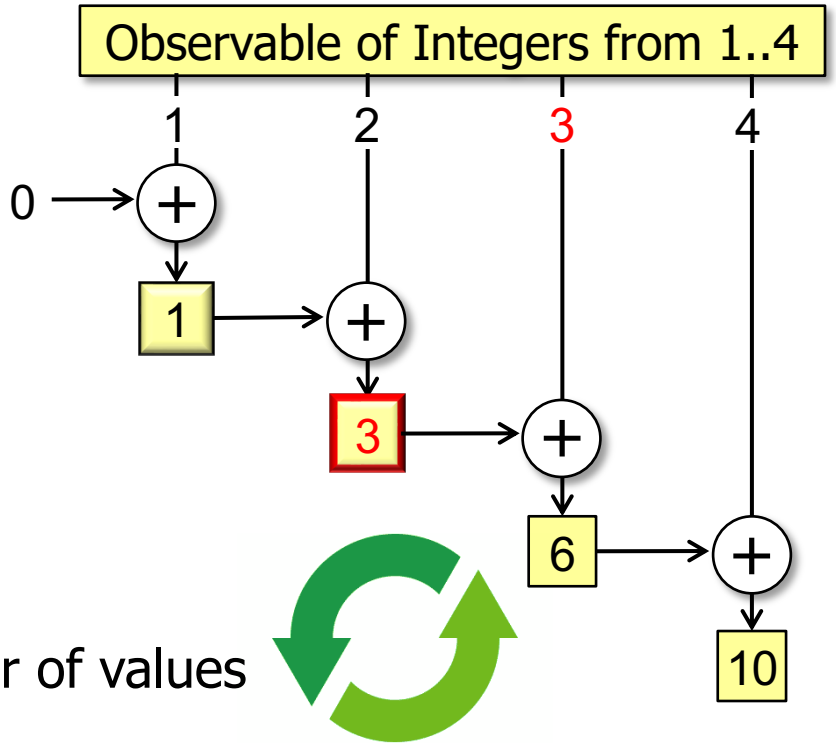
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

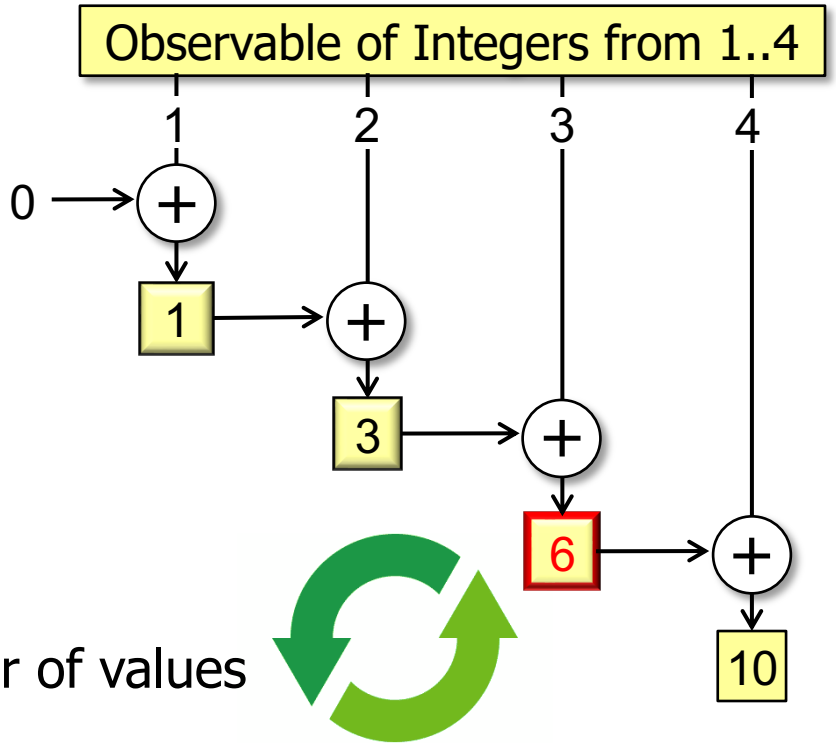
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

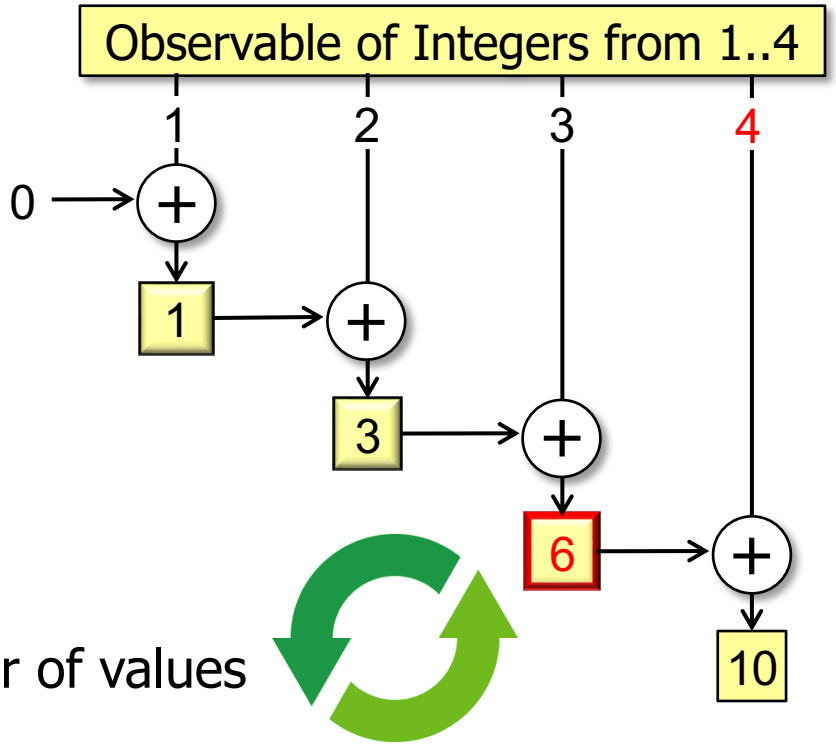
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

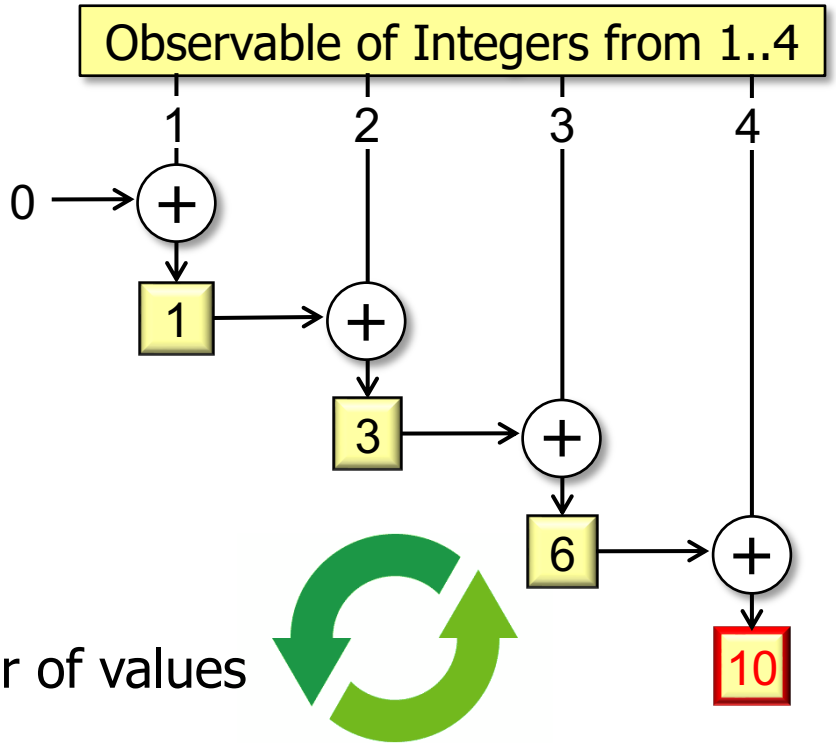
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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
      - This process repeats for each pair of values

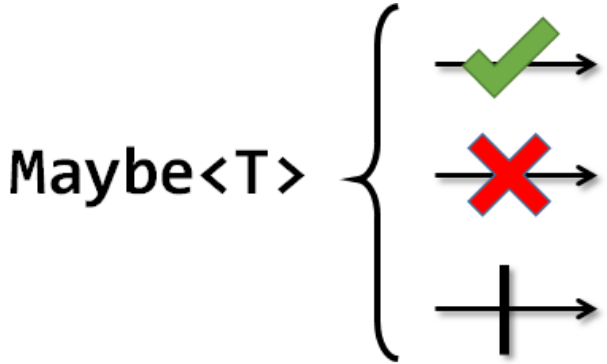
```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



Key Combining Operators in the Observable Class

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

```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```



See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/core/Maybe.html

Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The final result is emitted from the final call as the sole item of a Maybe
    - An empty Maybe will be returned if the Observable emits no items

```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```

Key Combining Operators in the Observable Class

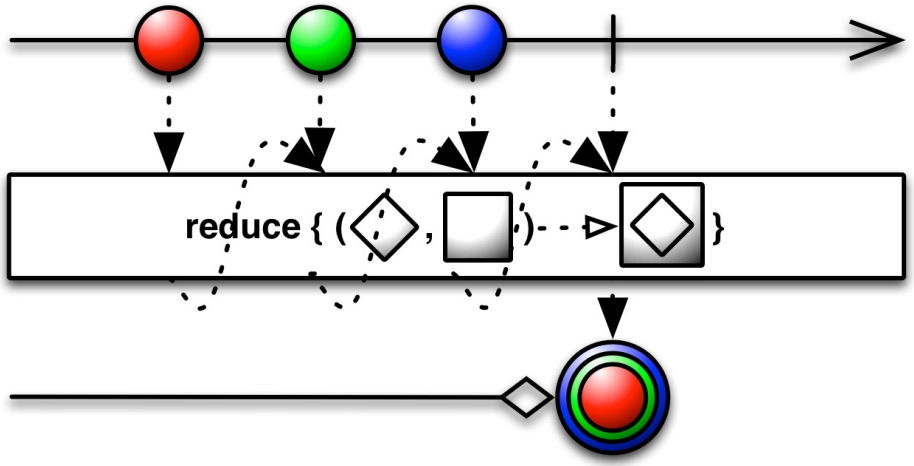
- The reduce() operator
  - Reduce this Observable's values into a single object of the same type as the emitted items
    - Reduction is performed using a BiFunction param
  - The final result is emitted from the final call as the sole item of a Maybe
    - An empty Maybe will be returned if the Observable emits no items

```
Maybe<T> reduce
   (BiFunction<T, T, T> reducer)
```

  - The internally accumulated value is discarded upon cancellation or error

Key Combining Operators in the Observable Class

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



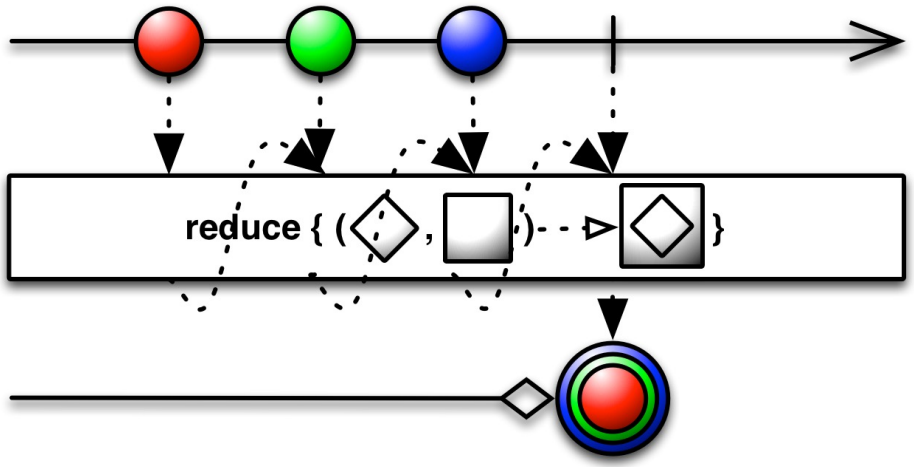

```
return Observable
   .fromArray(bigFractions)
   ...
   .flatMap(bf ->
            multiplyFractions(bf, Schedulers.computation()))
   .reduce(BigFraction::add)
   ...
```

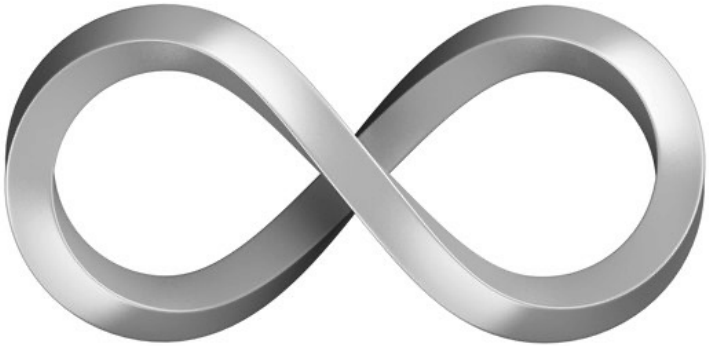
Sum the results of async multiplications

See Reactive/Observable/ex3/src/main/java/ObserableEx.java

Key Combining Operators in the Observable Class

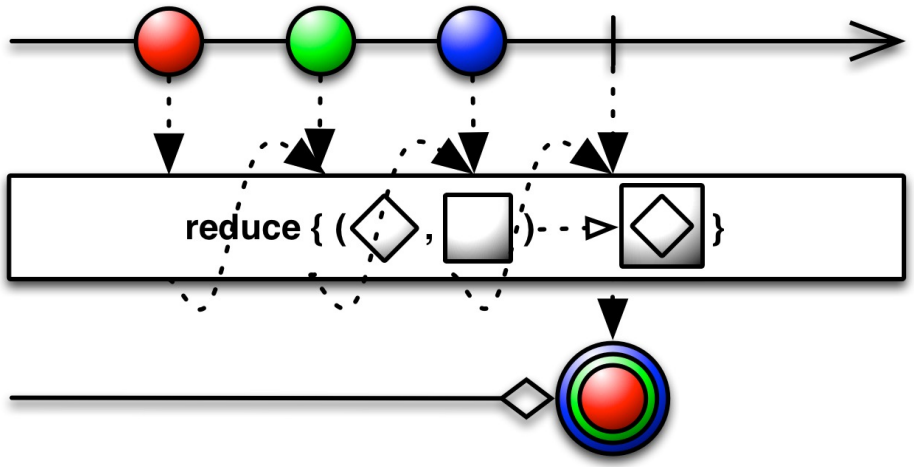
- The reduce() operator
  - Reduce this Observable's values into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
    - Sources that are infinite & never complete will never emit anything through this operator

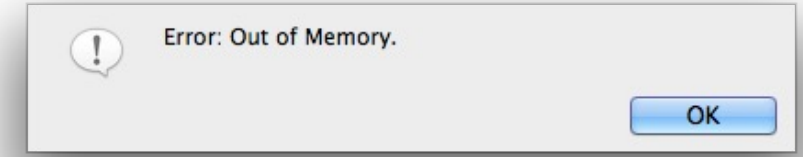




Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values 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

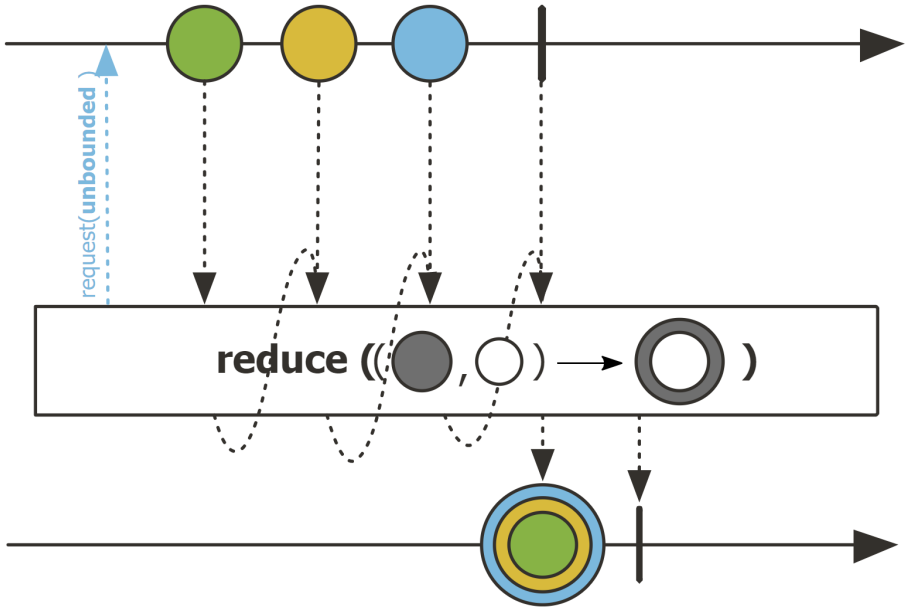




See docs.oracle.com/javase/8/docs/api/java/lang/OutOfMemoryError.html

Key Combining Operators in the Observable Class

- The reduce() operator
  - Reduce this Observable's values into a single object of the same type as the emitted items
  - Upstream must signal onComplete() before accumulator can be emitted
  - Project Reactor's Flux.reduce() operator works the same




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

Sum results of async multiplications

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

Key Combining Operators in the Observable Class

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

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();
```

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

Sum the List values

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

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