Key Combining Operators in the Observable Class (Part 3)

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
    - These operators create a Single by accumulating elements in an Observable stream
      - e.g., reduce(), collectInto(), & collect()

Key Combining Operators in the Observable Class

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure

```
Single<U> collectInto
   (U initialItem,
    BiConsumer<? super U, ? super T>
    collector)
```

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

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
    - The 1st param is the mutable data structure that accumulates (collects) the items

```
Single<U> collectInto
   (U initialItem,
    BiConsumer<? super U, ? super T>
    collector)

...
.collectInto
   (new ArrayList<BigFraction>(),
    List::add)
...
```

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
    - The 1st param is the mutable data structure that accumulates (collects) the items
  - The 2nd param is a BiConsumer that accepts the accumulator & an emitted item
    - The accumulator is modified accordingly

```
Single<U> collectInto
   (U initialItem,
    BiConsumer<? super U, ? super T>
    collector)

...
.collectInto
   (new ArrayList<BigFraction>(),
    List::add)
...
```

Interface BiConsumer<T1,T2>

Type Parameters:

T1 - the first value type

T2 - the second value type

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

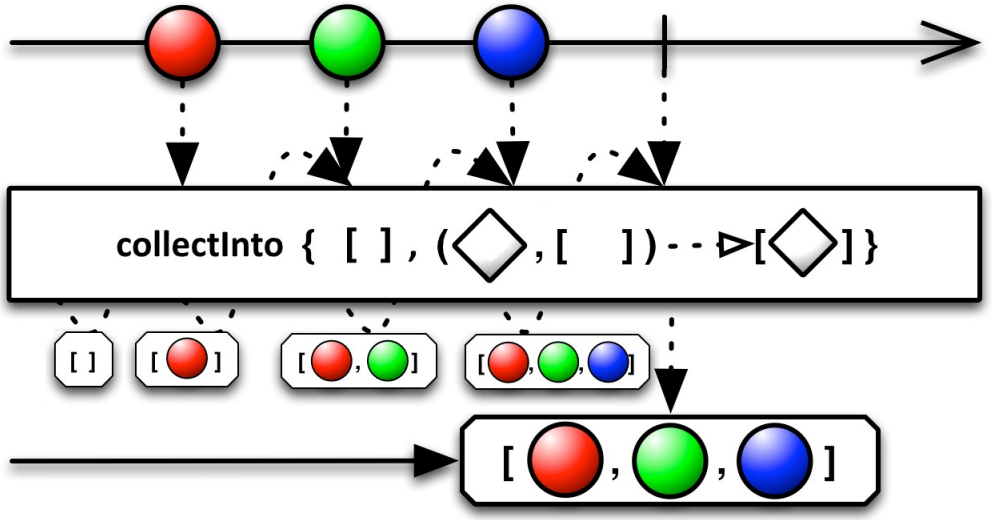
Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
    - The 1st param is the mutable data structure that accumulates (collects) the items
    - The 2nd param is a BiConsumer that accepts the accumulator & an emitted item
  - Returns a Single that emits the mutable data structure

```
Single<U> collectInto
   (U initialItem,
    BiConsumer<? super U, ? super T>
    collector)
```

Key Combining Operators in the Observable Class

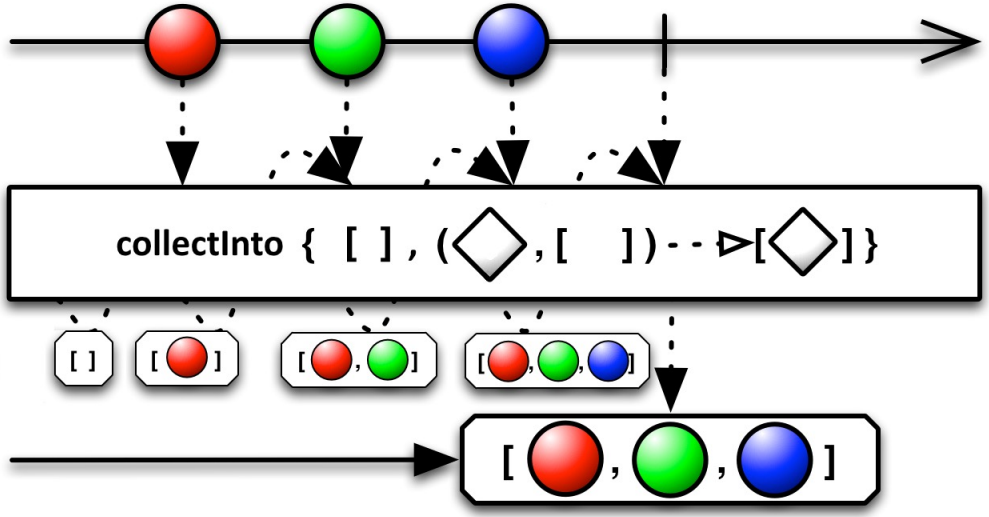
- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
  - This operator is a simplified version of reduce() that does not need to return the state on each pass



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

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
  - This operator is a simplified version of reduce() that does not need to return the state on each pass



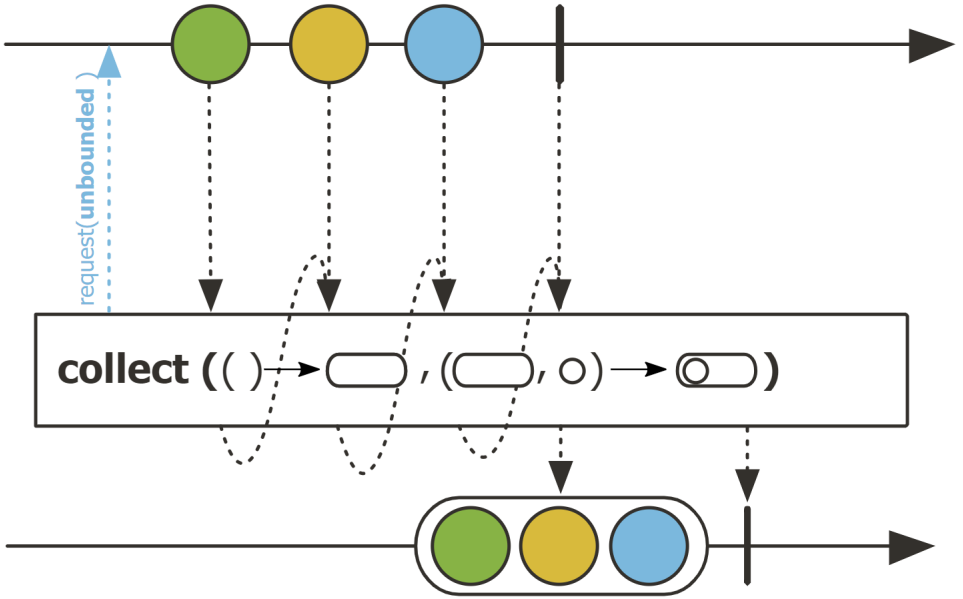

```
Observable
   .fromIterable(bigFractions)
   .flatMap(...)
   .filter(fraction -> fraction.compareTo(0) > 0)
   .collectInto(new ArrayList<BigFraction>(), List::add)
   ...
```

Collect filtered BigFractions into a list

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

Key Combining Operators in the Observable Class

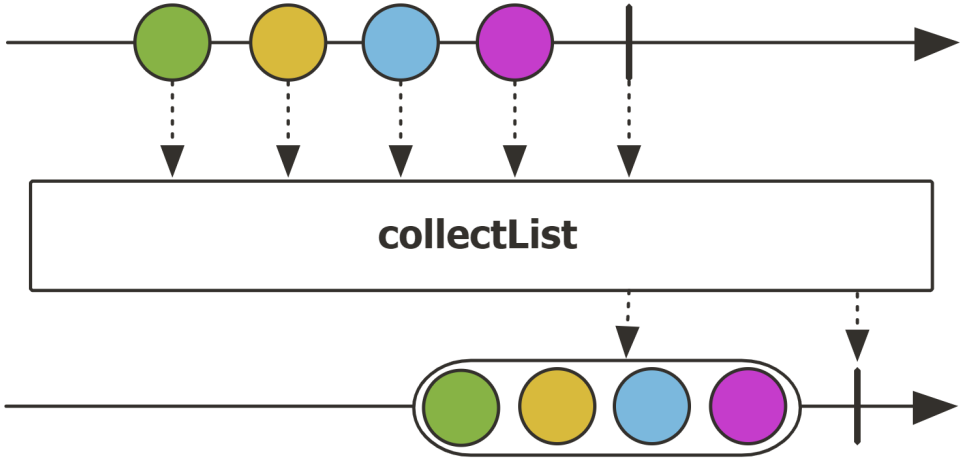
- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
  - Project Reactor's Flux.collect() operator works the same way



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

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
  - Project Reactor's Flux.collect() operator works the same way
    - Flux.collectList() is a more concise (albeit more limited) option



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

Key Combining Operators in the Observable Class

- The collectInto() operator
  - Collects items emitted by the finite source Observable into a single mutable data structure
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
  - Project Reactor's Flux.collect() operator 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.

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

Collect even #'d Integers into a List

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

Key Combining Operators in the Observable Class

- The collect() operator
  - Collects the finite upstream's values into a container

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

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

Key Combining Operators in the Observable Class

- The collect() operator
  - Collects the finite upstream's values into a container
    - The param is the Java Stream Collector interface defining the container supplier, accumulator, & finisher functions

```
<R, A> Single<U> collect
  (Collector<? super T,
             A,
             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 Observable Class

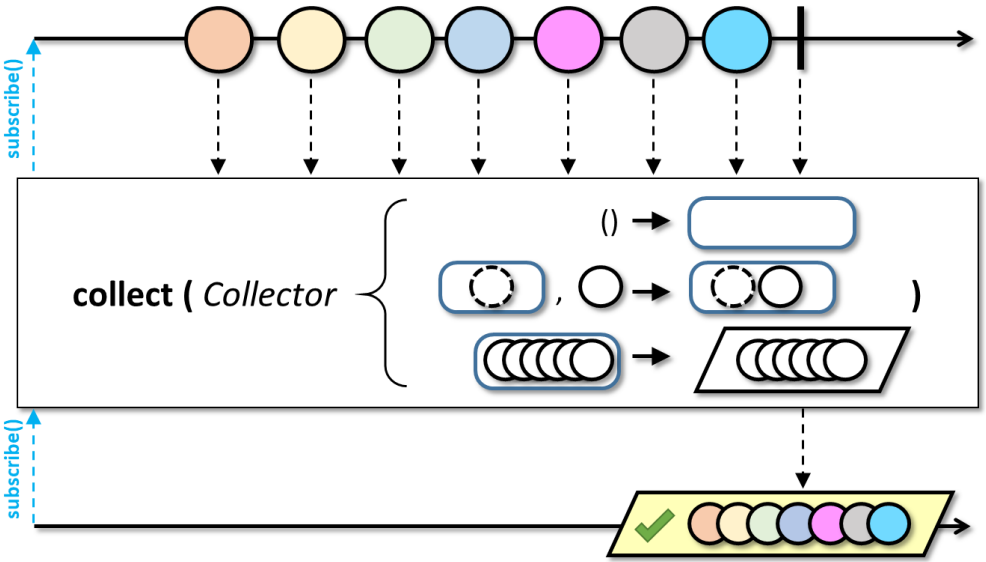
- The collect() operator
  - Collects the finite upstream's values into a container
    - The param is the Java Stream Collector interface defining the container supplier, accumulator, & finisher functions
  - Returns a Single that emits the container

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

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

Key Combining Operators in the Observable Class

- The collect() operator
  - Collects the finite upstream's values into a container
  - This operator is a simplified version of reduce() that does not need to return the state on each pass



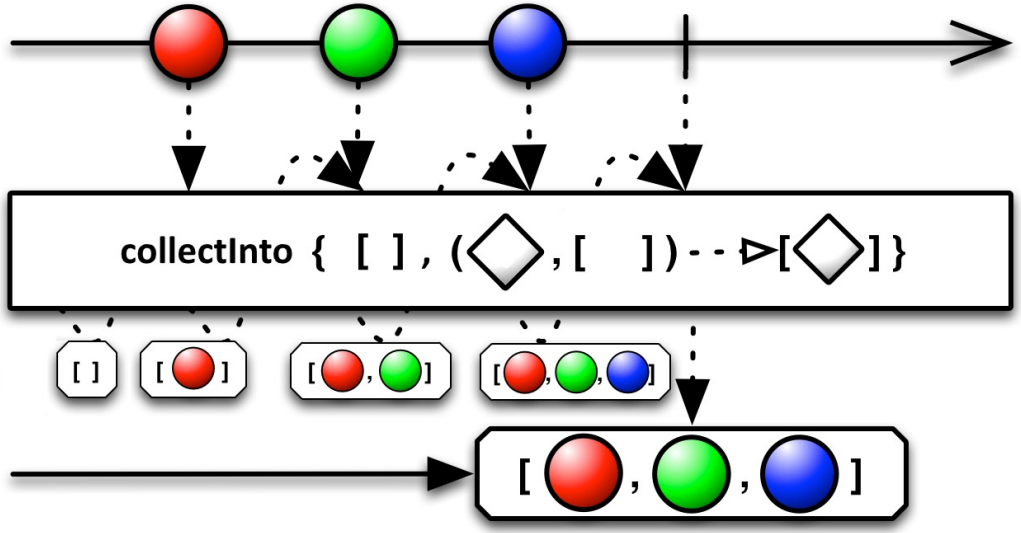

```
Observable
   .generate(emitter)
   .take(sMAX_FRACTIONS)
   .flatMap(...)
   .collect(toList())
   .flatMapCompletable(...);
```

Collect all the processed BigFractions into a List

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

Key Combining Operators in the Observable Class

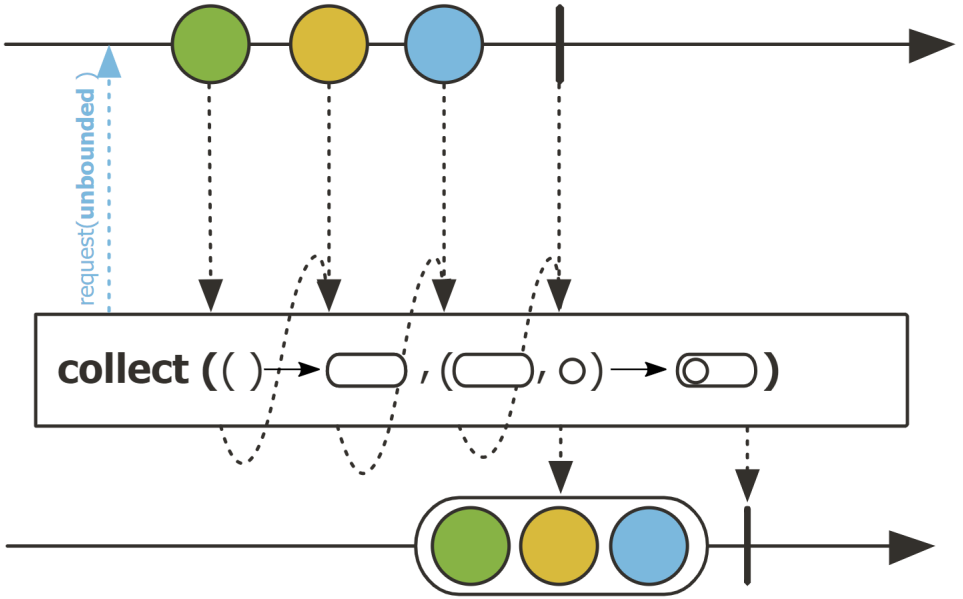
- The collect() operator
  - Collects the finite upstream's values into a container
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
    - It's also similar to operator Observable.collectInto()



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

Key Combining Operators in the Observable Class

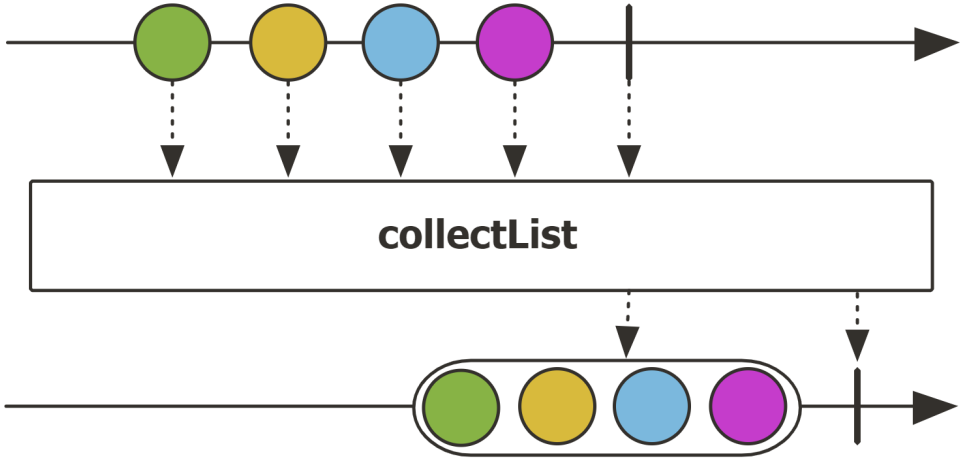
- The collect() operator
  - Collects the finite upstream's values into a container
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
  - Project Reactor's Flux.collect() operator works the same way



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

Key Combining Operators in the Observable Class

- The collect() operator
  - Collects the finite upstream's values into a container
  - This operator is a simplified version of reduce() that does not need to return the state on each pass



- Project Reactor's Flux.collect() operator works the same way
  - Flux.collectList() is a more concise (albeit more limited) option

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

Key Combining Operators in the Observable Class

- The collect() operator
  - Collects the finite upstream's values into a container
  - This operator is a simplified version of reduce() that does not need to return the state on each pass
  - Project Reactor's Flux.collect() operator 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

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

End of Key Combining Operators in the Observable Class (Part 3)