Overview of Key Classes in the RxJava API

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

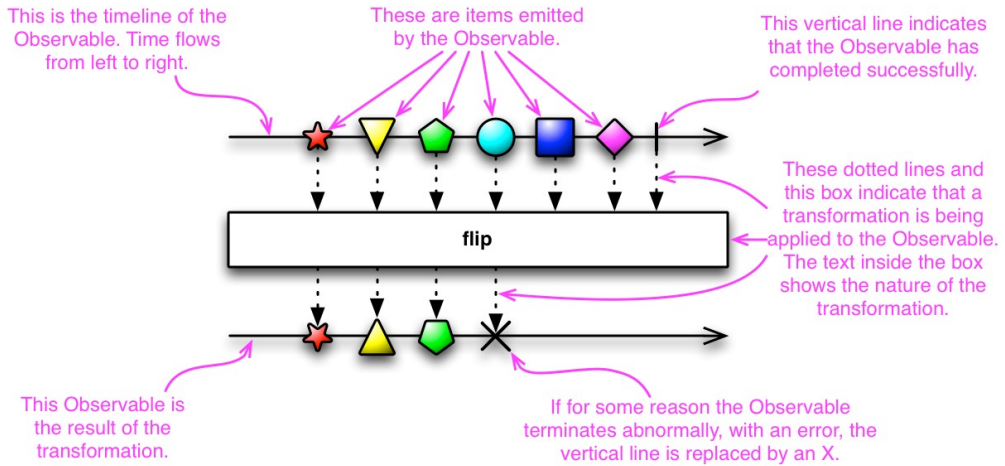
Professor of Computer Science

Institute for Software Integrated Systems

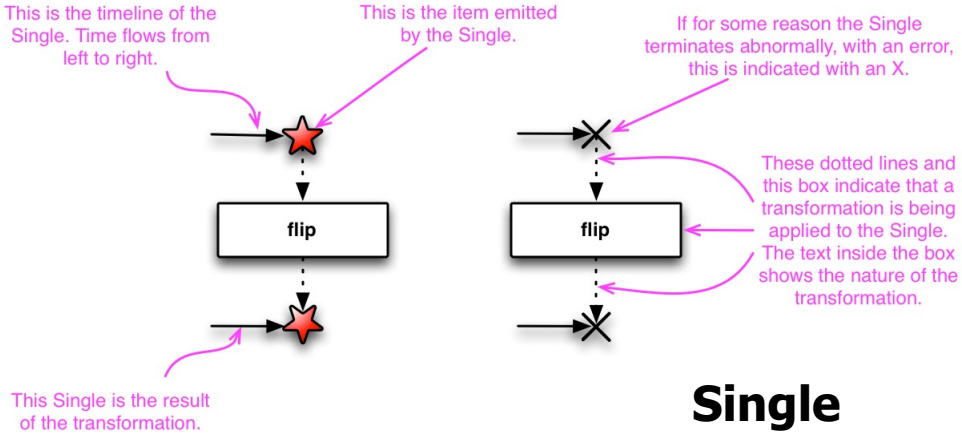
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand key classes in the RxJava API
- Understand key classes in the RxJava API



Flowable & Observable



Single

Key Classes in the RxJava API

Key Classes in the RxJava API

- RxJava has three key classes

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value

Class Single<T>

java.lang.Object
    io.reactivex.rxjava3.core.Single<T>

Type Parameters:
    `T` - the type of the item emitted by the `Single`

All Implemented Interfaces:
    SingleSource<T>

Direct Known Subclasses:
    SingleSubject

```
public abstract class Single<T>
extends Object
implements SingleSource<T>
```

The `Single` class implements the Reactive Pattern for a single value response.

`Single` behaves similarly to `Observable` except that it can only emit either a single successful value or an error (there is no `onComplete` notification as there is for an `Observable`).

The `Single` class implements the `SingleSource` base interface and the default consumer type it interacts with is the `SingleObserver` via the `subscribe(SingleObserver)` method.

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

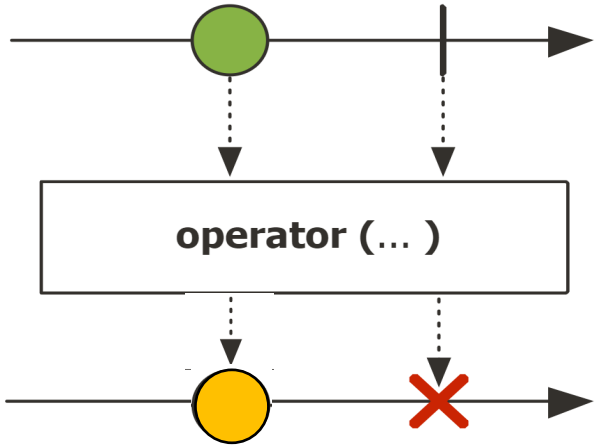
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>

```
BigFraction unreducedFraction =
  makeBigFraction(...);

Single
   .fromCallable(() -> BigFraction
        .reduce(unreducedFraction))
   .subscribeOn
       (Schedulers.single())
   .map(result ->
        result.toMixedString())
   .doOnSuccess(result ->
                System.out.println
                 ("big fraction = "
                 + result + "\n"));
```

Key Classes in the RxJava API

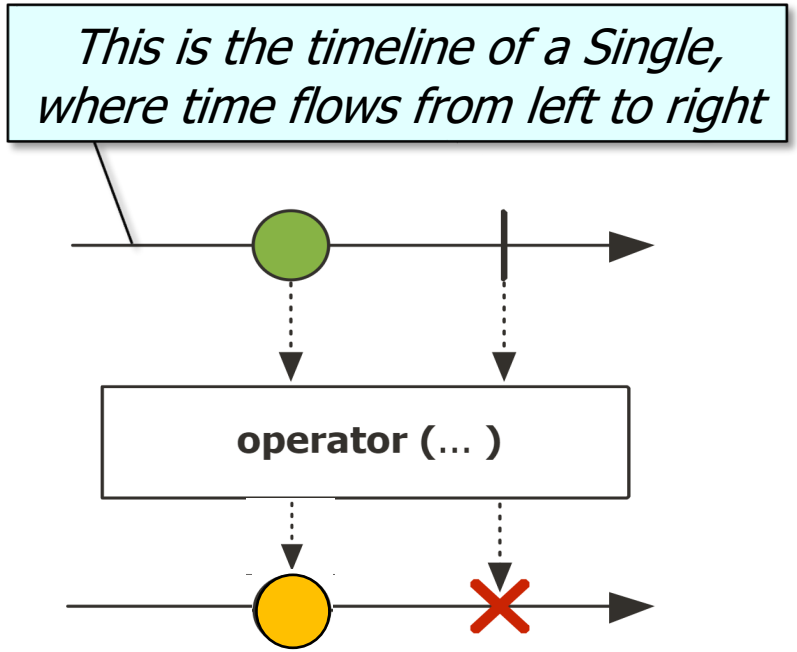
- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



See medium.com/@jshvarts/read-marble-diagrams-like-a-pro-3d72934d3ef5

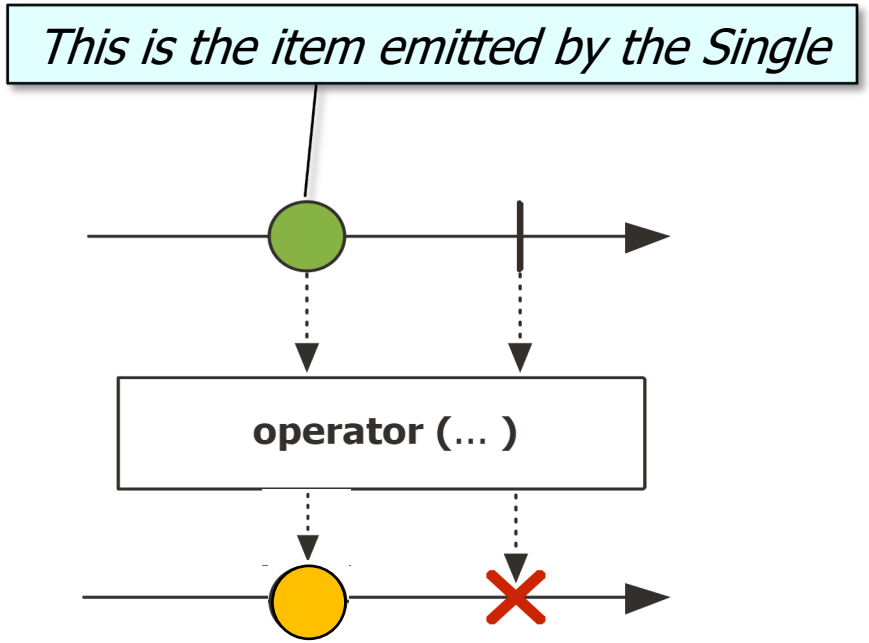
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



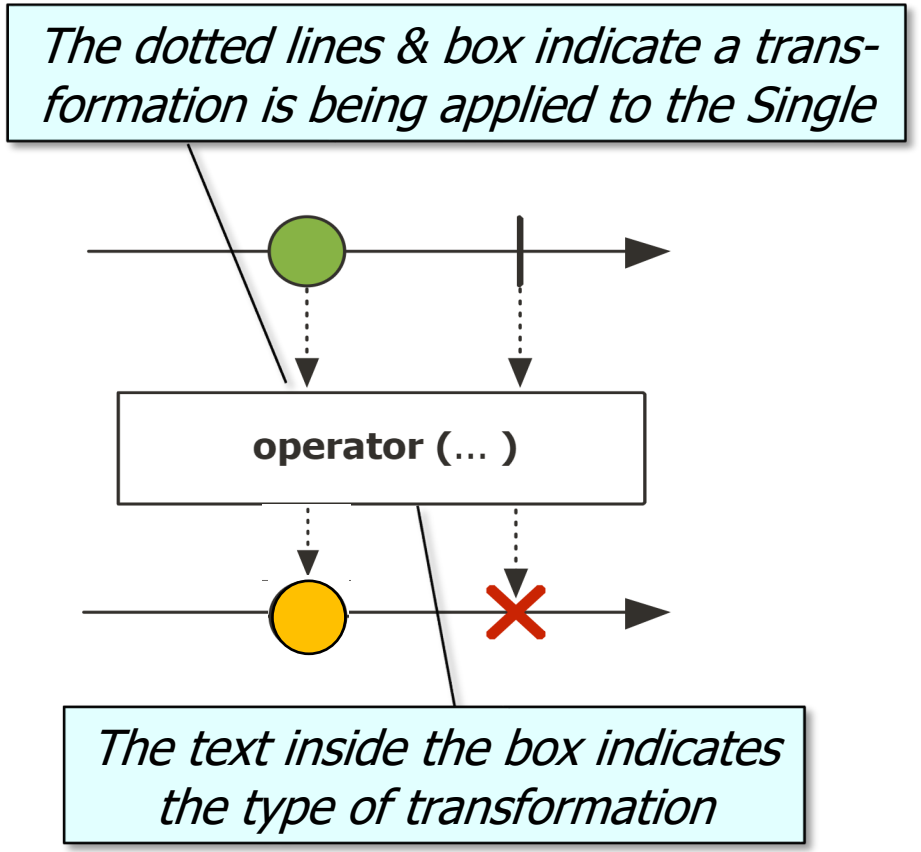
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



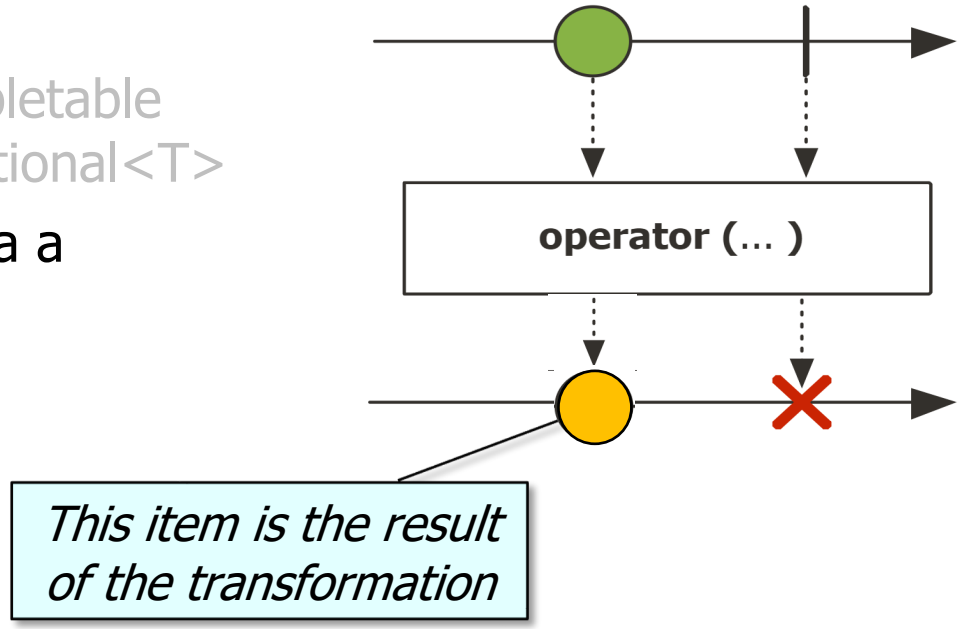
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



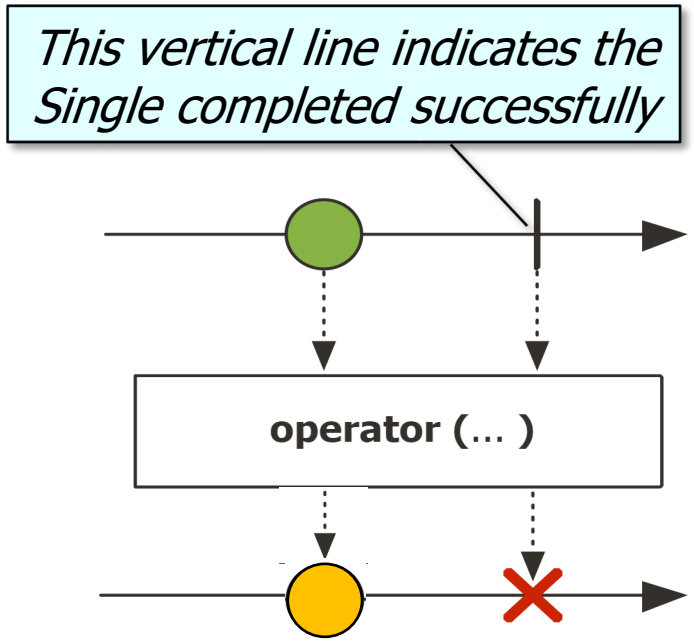
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



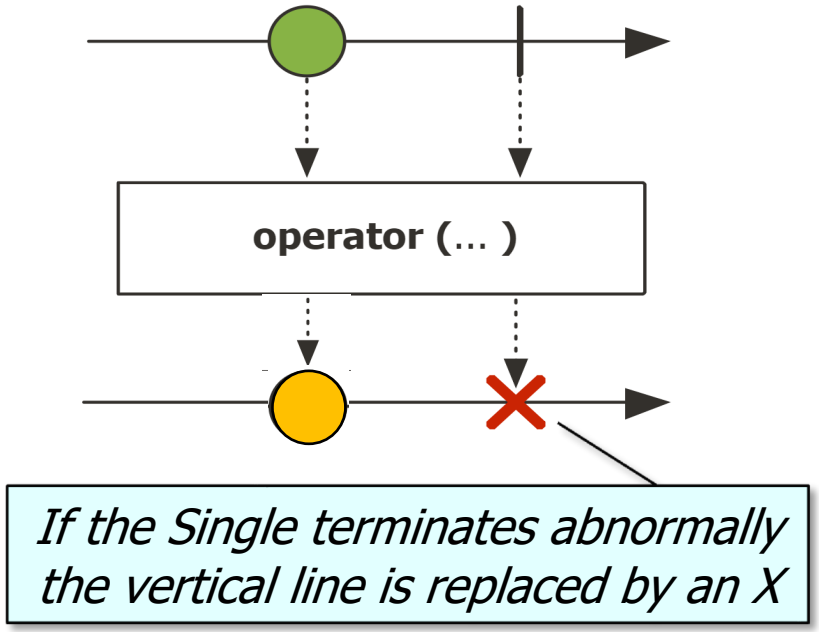
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"



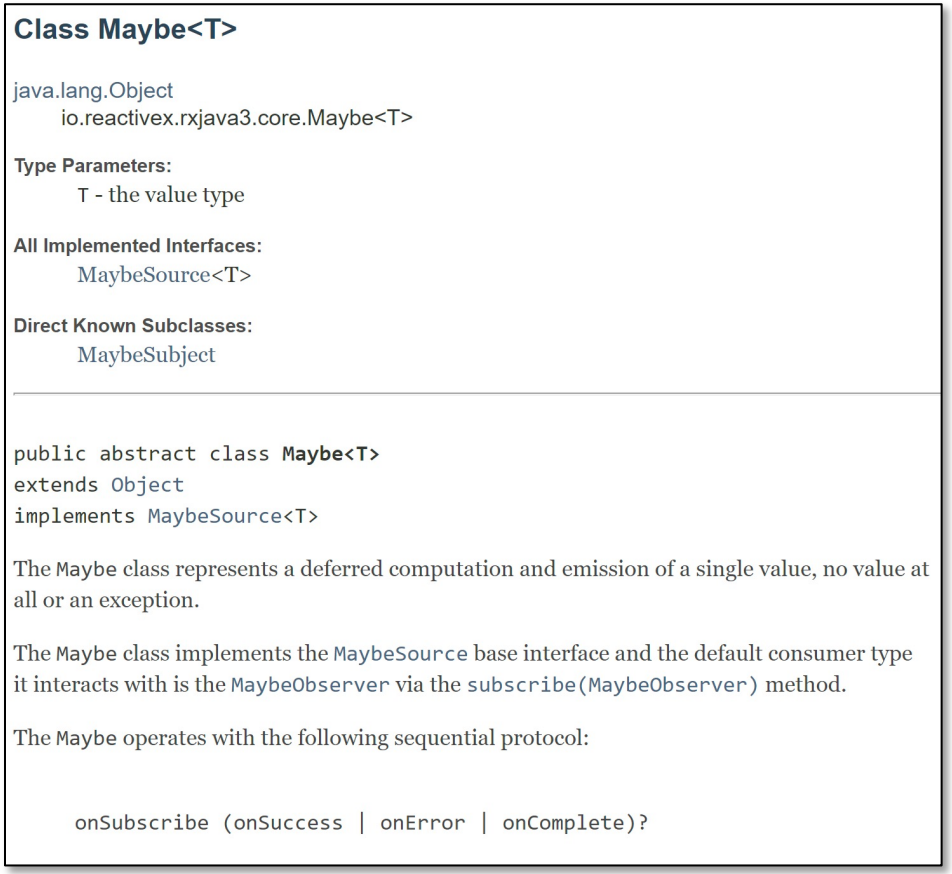
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"
    - Provides many operators

- Factory method operators
- Transforming operators
- Action operators
- Concurrency & scheduler operators
- Combining operators
- Suppressing operators
- Blocking operators
- etc.

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"
    - Provides many operators
  - Maybe is a variant of Single



Class Maybe<T>

java.lang.Object
io.reactivex.rxjava3.core.Maybe<T>

Type Parameters:
T - the value type

All Implemented Interfaces:
MaybeSource<T>

Direct Known Subclasses:
MaybeSubject

```
public abstract class Maybe<T>
extends Object
implements MaybeSource<T>
```

The Maybe class represents a deferred computation and emission of a single value, no value at all or an exception.

The Maybe class implements the MaybeSource base interface and the default consumer type it interacts with is the MaybeObserver via the subscribe(MaybeObserver) method.

The Maybe operates with the following sequential protocol:

```
onSubscribe (onSuccess | onError | onComplete)?
```

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

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
    - Similar to a Java Completable Future or an async Optional<T>
    - Can be documented via a "marble diagram"
    - Provides many operators
  - Maybe is a variant of Single
    - It may emit a single value, no value at all, or an exception

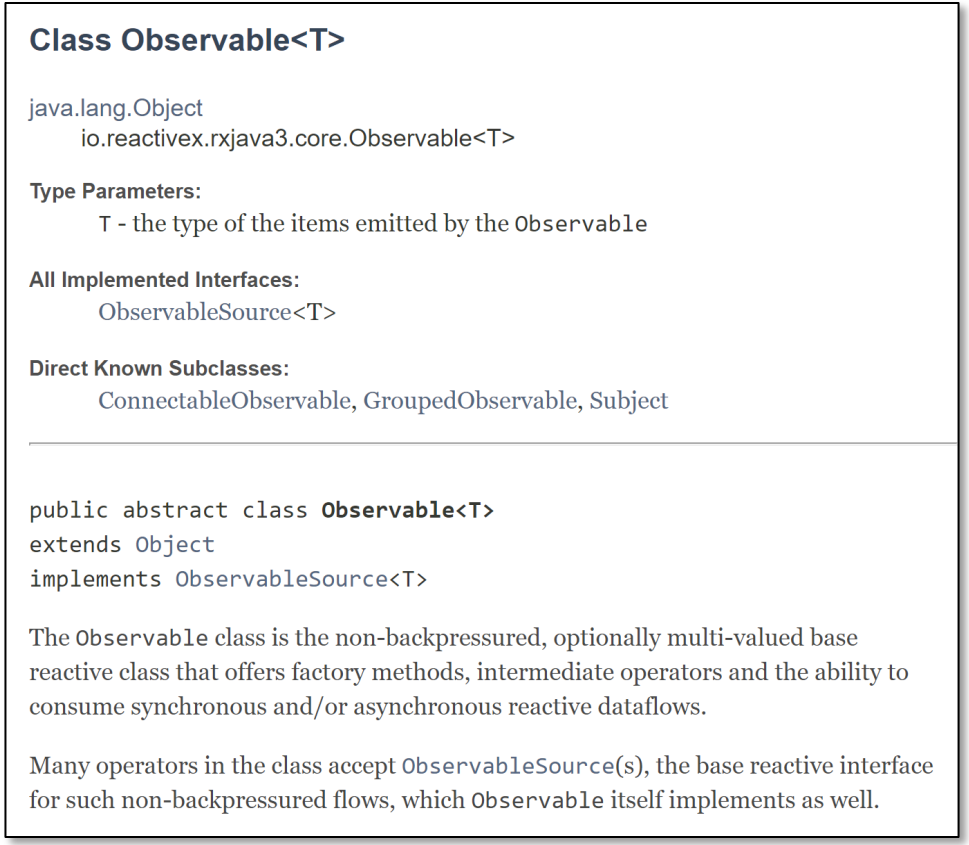
```
BigInteger factorial(BigInteger n){
  return Observable
    .rangeLong(1, n.longValue())
    .map(BigInteger::valueOf)
    .reduce(BigInteger::multiply)
```

reduce() returns a Maybe, which may contain no value at all if n is 0

```
    .blockingGet(BigInteger.ONE);
```

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail



Class Observable<T>

java.lang.Object
  io.reactivex.rxjava3.core.Observable<T>

Type Parameters:
  `T` - the type of the items emitted by the `Observable`

All Implemented Interfaces:
  ObservableSource<T>

Direct Known Subclasses:
  ConnectableObservable, GroupedObservable, Subject

```
public abstract class Observable<T>
extends Object
implements ObservableSource<T>
```

The `Observable` class is the non-backpressured, optionally multi-valued base reactive class that offers factory methods, intermediate operators and the ability to consume synchronous and/or asynchronous reactive dataflows.

Many operators in the class accept `ObservableSource`(s), the base reactive interface for such non-backpressured flows, which `Observable` itself implements as well.

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

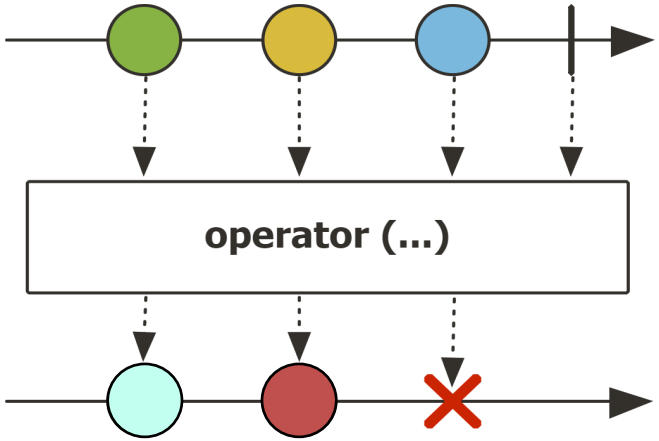
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
      - i.e., completable futures used with a Java stream

```
return Observable
   .fromArray(bigFractions)

   .subscribeOn(scheduler)

   .flatMap(reducedFraction ->
       Observable
          .fromCallable(() ->
            reducedFraction.multiply
               (sBigReducedFraction))

          .subscribeOn
             (scheduler))

   .reduce(BigFraction::add);
```

Key Classes in the RxJava API

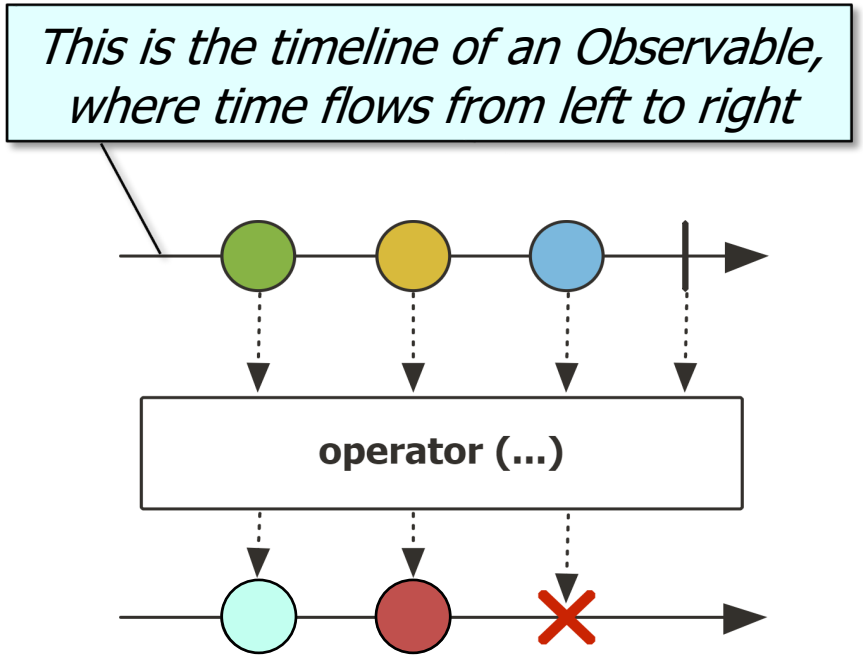
- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



See medium.com/@jshvarts/read-marble-diagrams-like-a-pro-3d72934d3ef5

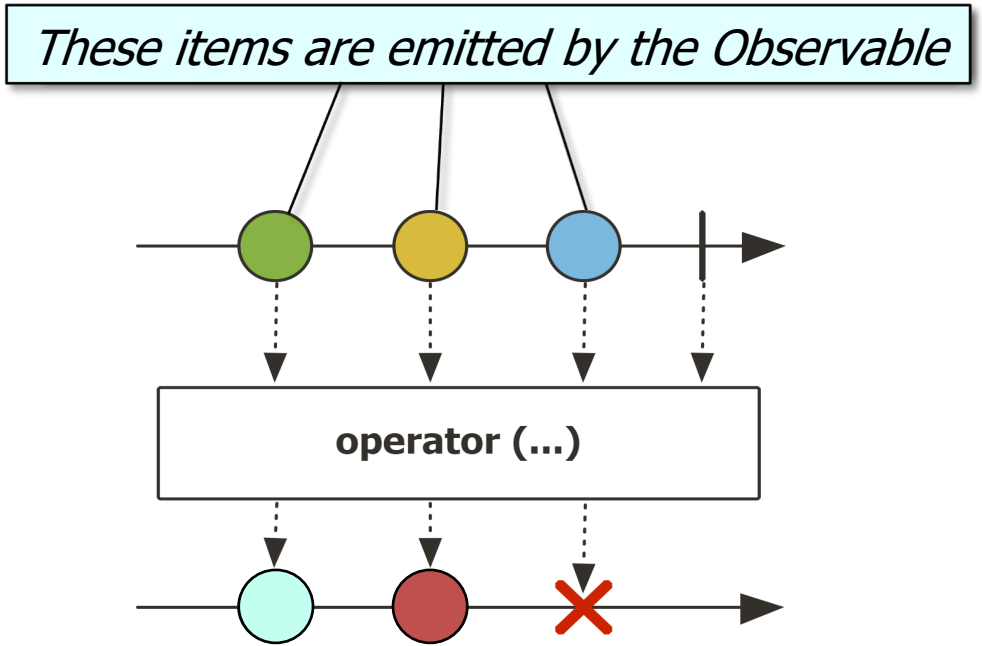
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



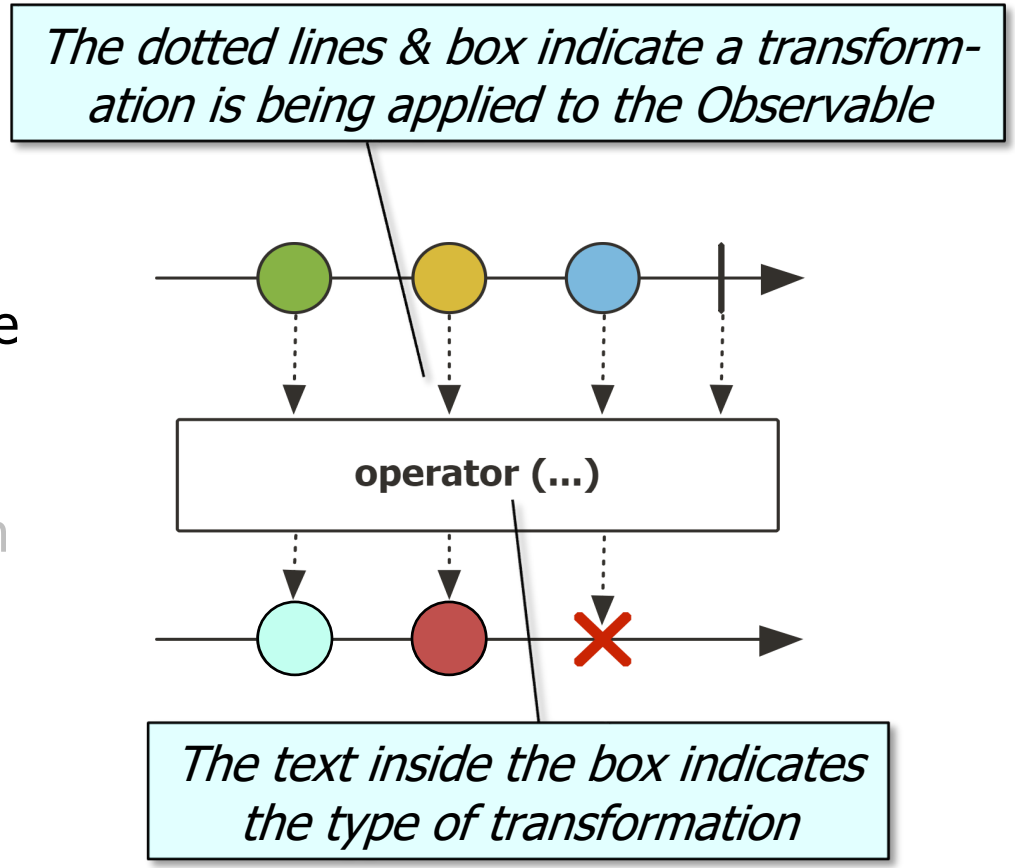
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



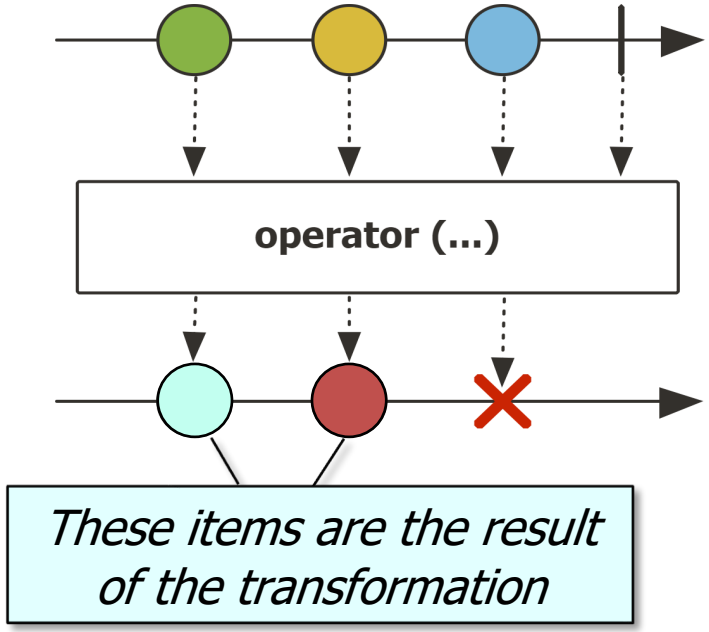
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



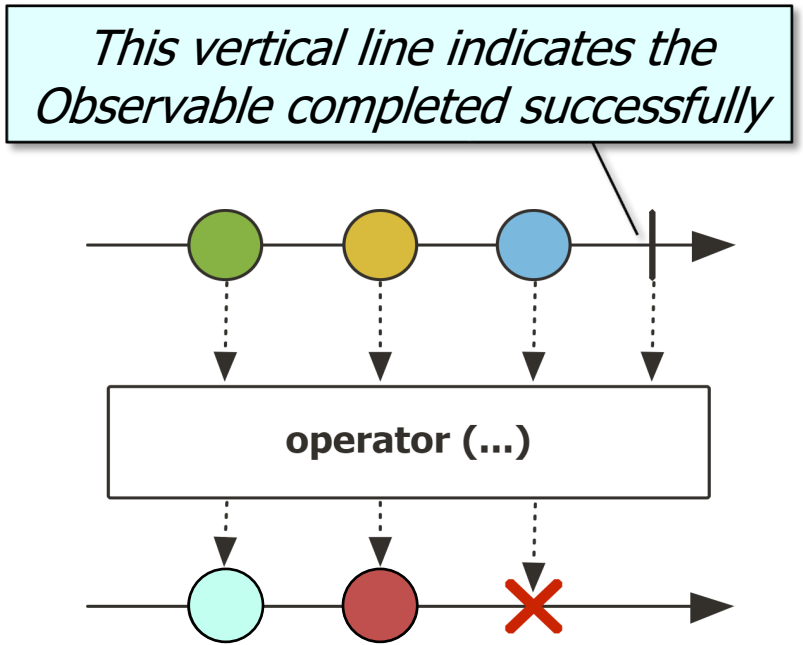
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



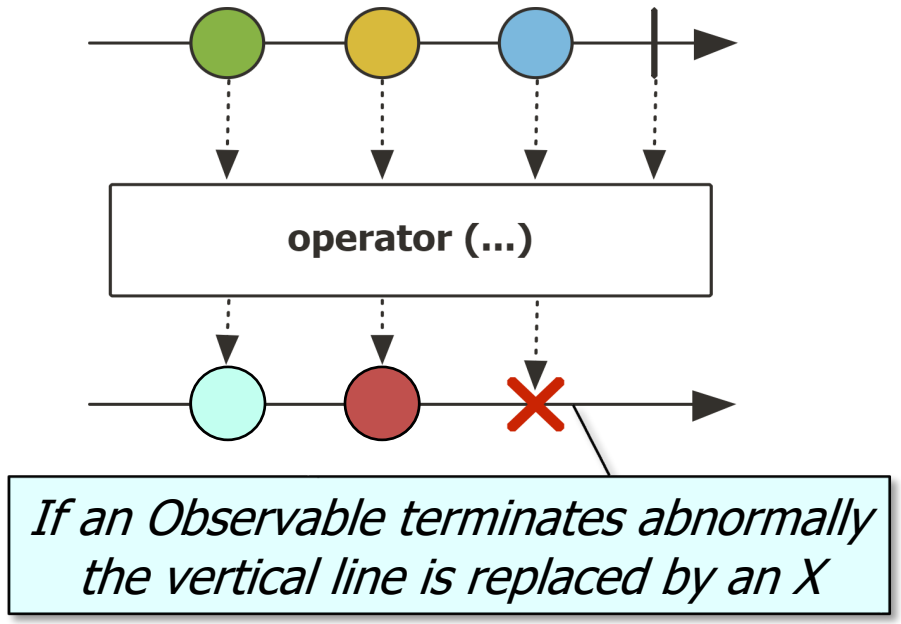
Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram



Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram

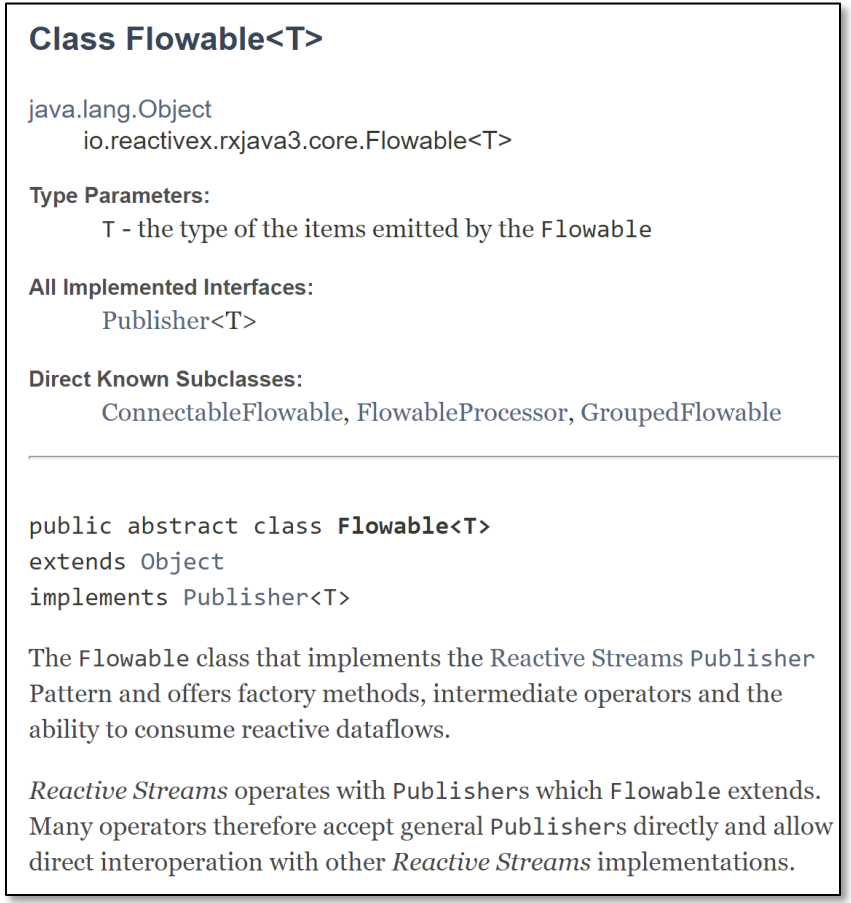


Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
    - Similar to an async Java stream
    - Can also be documented via a marble diagram
    - Provides many operators
      - Factory method operators
      - Transforming operators
      - Action operators
      - Concurrency & scheduler operators
      - Combining operators
      - Terminal operators
      - Suppressing operators
      - Blocking operators
      - etc.

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
  - **Flowable** – Generalizes Observable to support backpressure



Class Flowable<T>

java.lang.Object
io.reactivex.rxjava3.core.Flowable<T>

Type Parameters:
T - the type of the items emitted by the Flowable

All Implemented Interfaces:
Publisher<T>

Direct Known Subclasses:
ConnectableFlowable, FlowableProcessor, GroupedFlowable

```
public abstract class Flowable<T>
extends Object
implements Publisher<T>
```

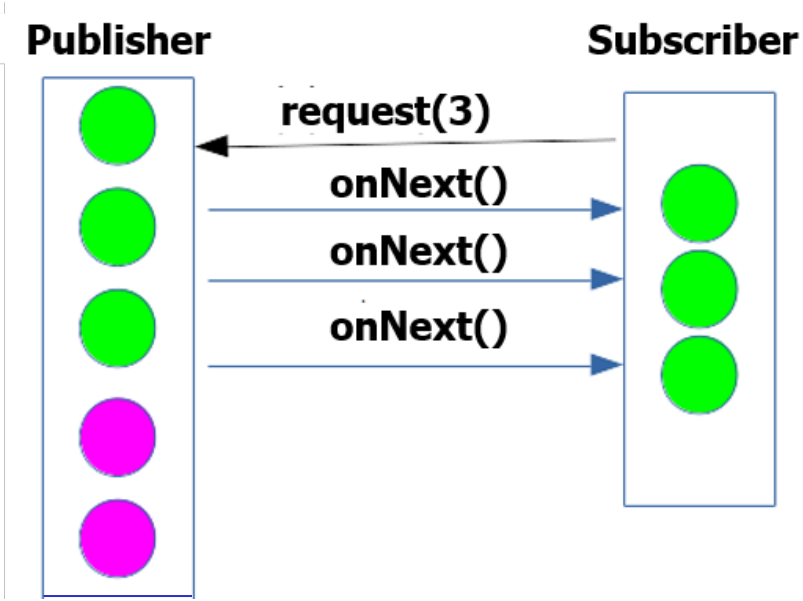
The Flowable class that implements the Reactive Streams Publisher Pattern and offers factory methods, intermediate operators and the ability to consume reactive dataflows.

Reactive Streams operates with Publishers which Flowable extends. Many operators therefore accept general Publishers directly and allow direct interoperation with other *Reactive Streams* implementations.

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

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
  - **Flowable** – Generalizes Observable to support backpressure
    - The subscriber indicates to the publisher how much data it can consume



See www.baeldung.com/rxjava-backpressure

Key Classes in the RxJava API

- RxJava has three key classes
  - **Single** – Completes successfully or with failure, may or may not emit a single value
  - **Observable** – Emits an indefinite # of events (zero to infinite) & may complete successfully or fail
  - **Flowable** – Generalizes Observable to support backpressure
    - The subscriber indicates to the publisher how much data it can consume
    - A Flowable can be converted to a ParallelFlowable
      - ParallelFlowable can operate on multiple streams of data concurrently

```
return Flowable
   .fromArray(bigFractions)
   .parallel()
   .runOn(scheduler)
   .flatMap(bigFraction ->
      bigFraction.multiply
         (sBigReducedFraction))
   .sequential()
   .reduce(BigFraction::add)
   ...
```

See reactivex.io/RxJava/3.x/javadoc/io/reactivex/rxjava3/parallel/ParallelFlowable.html

End of Overview of Key Classes in the RxJava API