Key Error Handling Operators in the Flux Class

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
  - Concurrency & scheduler operators
  - Error handling operators
    - These operators handle errors that occur in a Flux chain
      - e.g., onErrorContinue(), onErrorResume(), & onErrorStop()

Key Error Handling Operators in the Flux Class

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element

```
Flux<T> onErrorContinue
   (BiConsumer<Throwable, Object>
      errorConsumer)
```

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

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
    - The param is a BiConsumer that is fed with errors matching the predicate & the value that triggered the error

```
Flux<T> onErrorContinue
   (BiConsumer<Throwable, Object>
      errorConsumer)
```

Interface BiConsumer<T,U>

Type Parameters:

T - the type of the first argument to the operation

U - the type of the second argument to the operation

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/BiConsumer.html

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
    - The param is a BiConsumer that is fed with errors matching the predicate & the value that triggered the error
      - The type of the error is a subclass of Throwable

```
Flux<T> onErrorContinue
   (BiConsumer<Throwable, Object>
      errorConsumer)
```

```
public class Throwable
extends Object
implements Serializable
```

The `Throwable` class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java `throw` statement. Similarly, only this class or one of its subclasses can be the argument type in a `catch` clause. For the purposes of compile-time checking of exceptions, `Throwable` and any subclass of `Throwable` that is not also a subclass of either `RuntimeException` or `Error` are regarded as checked exceptions.

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

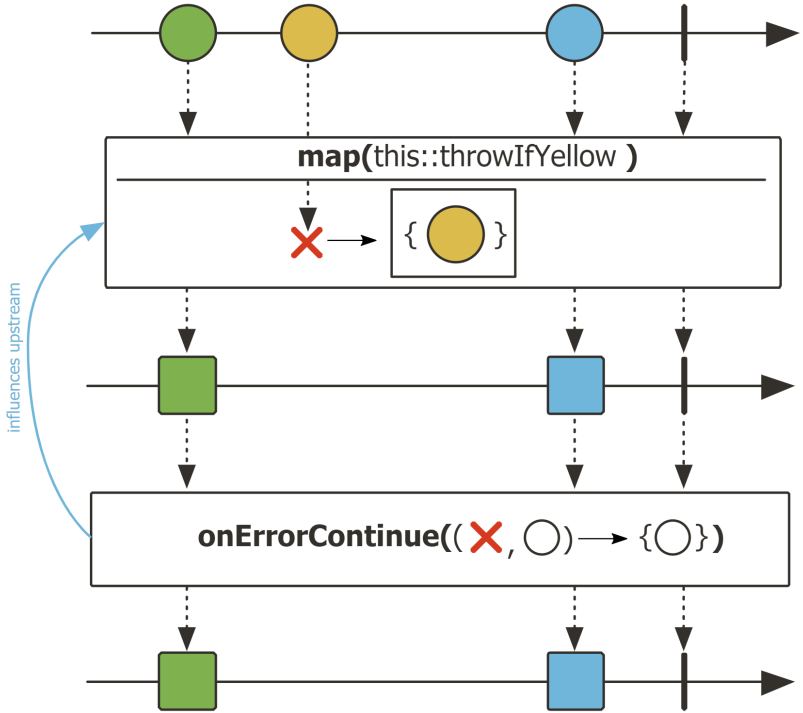
Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
    - The param is a BiConsumer that is fed with errors matching the predicate & the value that triggered the error
    - Returns a Flux that attempts to continue processing when errors (exceptions) occur

```
Flux<T> onErrorContinue
   (BiConsumer<Throwable, Object>
      errorConsumer)
```

Key Error Handling Operators in the Flux Class

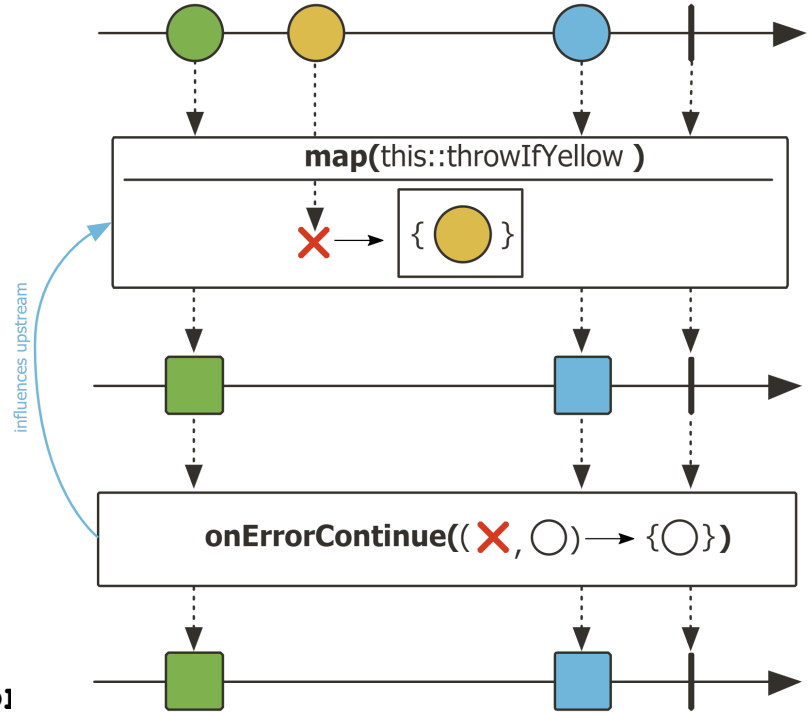
- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further



See en.wikipedia.org/wiki/Error_hiding

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further




```
return Flux
   .fromIterable(denominators)
   .map(denominator -> BigFractio
         .valueOf(Math.abs(sRANDOM.nextInt()), denominator))
   .onErrorContinue(logErrorAndContinue)
   ...
```

Continue processing

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

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
    - It also affects the behavior of onErrorResume() operators..

onErrorResume() is ignored if onErrorContinue() appears downstream

```
Flux
   .range(1,5)
   .doOnNext(i -> log("i = " + i))
   .map(i -> i == 2 ? i / 0 : i)
   .map(i -> i * 2)
   .onErrorResume(err -> {
      log("resuming");
      return Flux.empty();
    })
   .onErrorContinue((err, i) ->
       {log("continuing={}", i);})
   .reduce(Math::addExact)
   .doOnNext(i ->
             println("sum=" + i))
   .block();
```

See devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
    - It also affects the behavior of onErrorResume() operators..
      - See upcoming discussion of onErrorStop() for a solution

onErrorStop

```
public final Flux<T> onErrorStop()
```

If an `onErrorContinue(BiConsumer)` variant has been used downstream, reverts to the default 'STOP' mode where errors are terminal events upstream. It can be used for easier scoping of the on next failure strategy or to override the inherited strategy in a sub-stream (for example in a flatMap). It has no effect if `onErrorContinue(BiConsumer)` has not been used downstream.

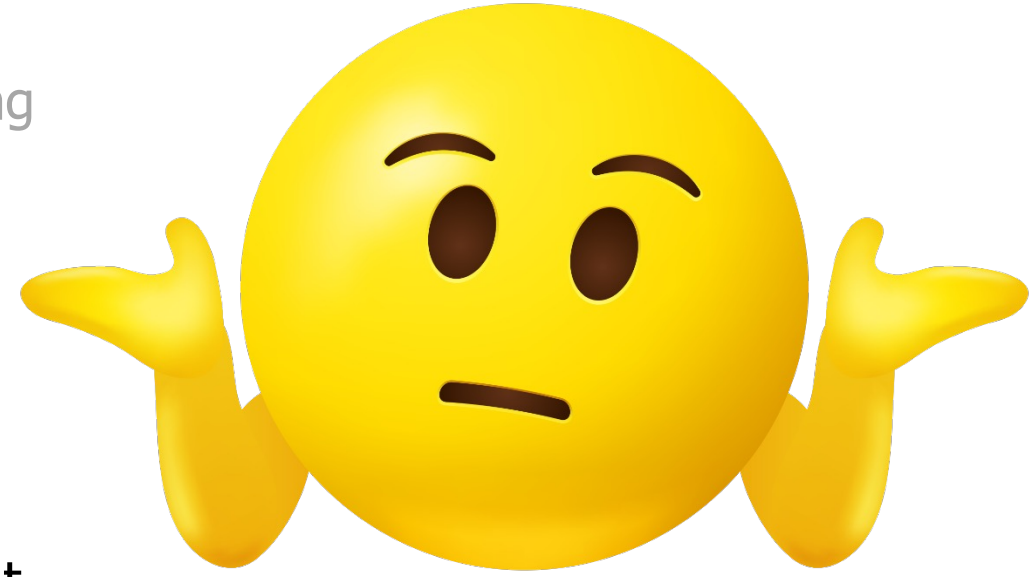
Returns:

a `Flux` that terminates on errors, even if `onErrorContinue(BiConsumer)` was used downstream

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

Key Error Handling Operators in the Flux Class

- The onErrorContinue() operator
  - Recovers from errors by dropping the incriminating element from the sequence & continuing with subsequent element
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
  - RxJava's has no direct equivalent



Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs

```
Flux<T> onErrorResume
  (Function<? super Throwable,
            ? extends Publisher
                  <? extends T>>
   fallback)
```

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

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
    - The param is a Function that chooses the fallback, depending on the type of the error

```
Flux<T> onErrorResume
  (Function<? super Throwable,
            ? extends Publisher
                  <? extends T>>
   fallback)
```

Interface Function<T,R>

Type Parameters:

T - the type of the input to the function

R - the type of the result of the function

All Known Subinterfaces:

UnaryOperator<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/Function.html

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
    - The param is a Function that chooses the fallback, depending on the type of the error
      - The type of the error is a subclass of Throwable

```
Flux<T> onErrorResume
  (Function<? super Throwable,
            ? extends Publisher
                  <? extends T>>
   fallback)
```

public class **Throwable**
extends Object
implements Serializable

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement. Similarly, only this class or one of its subclasses can be the argument type in a catch clause. For the purposes of compile-time checking of exceptions, Throwable and any subclass of Throwable that is not also a subclass of either RuntimeException or Error are regarded as checked exceptions.

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

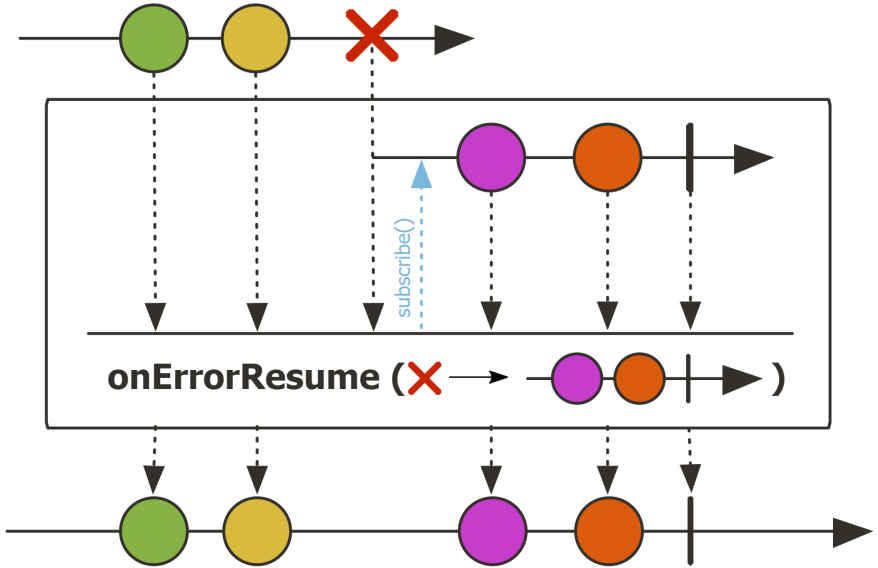
Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
    - The param is a Function that chooses the fallback, depending on the type of the error
  - Returns a Flux that falls back to the publisher when an onError() occurs

```
Flux<T> onErrorResume
  (Function<? super Throwable,
            ? extends Publisher
                  <? extends T>>
   fallback)
```

Key Error Handling Operators in the Flux Class

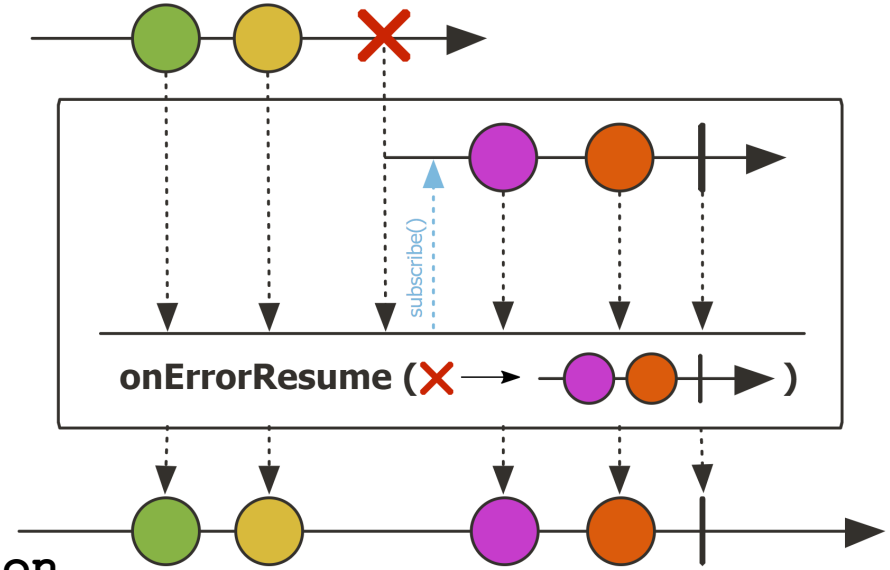
- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further



See en.wikipedia.org/wiki/Error_hiding

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further



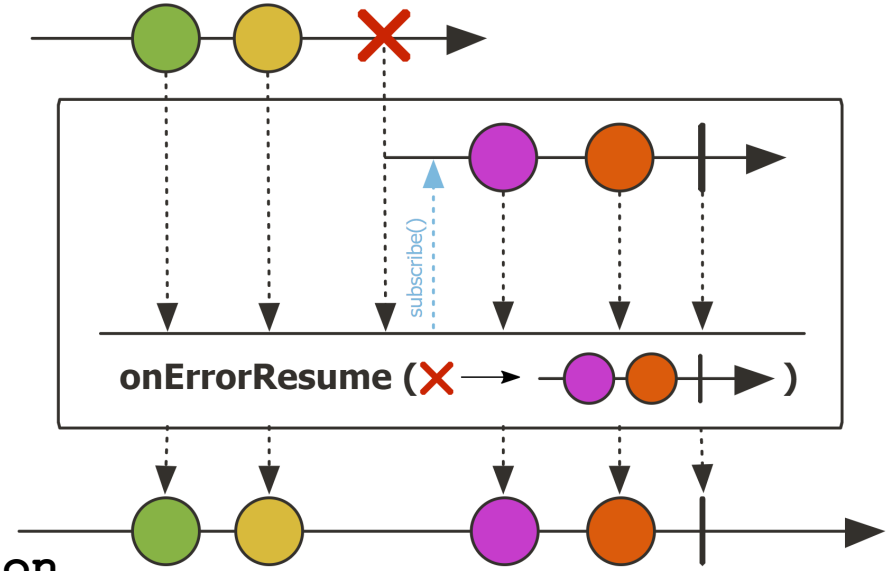

```
return Flux
   .fromIterable(denominators)
   .map(denominator -> BigFraction
         .valueOf(Math.abs(sRANDOM.nextInt()), denominator))
   .onErrorResume(__ -> Flux.empty())
   .onErrorStop()
   .collectList()
   ...
```

Convert Arithmetic Exception to empty Flux

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

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further




```
return Flux
   .fromIterable(denominators)
   .map(denominator -> BigFraction
        .valueOf(Math.abs(sRANDOM.nextInt()), denominator))
   .onErrorResume(__ -> Flux.empty())
   .onErrorStop()
   .collectList()
   ...
```

Stop processing in this Flux stream when an error occurs

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

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
    - Beware when onErrorResume() is used in conjunction with onErrorContinue()

onErrorResume() is ignored if onErrorContinue() appears downstream

```
Flux
   .range(1,5)
   .doOnNext(i -> log("i = " + i))
   .map(i -> i == 2 ? i / 0 : i)
   .map(i -> i * 2)
   .onErrorResume(err -> {
      log("resuming");
      return Flux.empty();
    })
   .onErrorContinue((err, i) ->
       {log("continuing={}", i);})
   .reduce(Math::addExact)
   .doOnNext(i ->
             println("sum=" + i))
   .block();
```

See devdojo.com/ketonemaniac/reactor-onerrorcontinue-vs-onerrorresume

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
    - Beware when onErrorResume() is used in conjunction with onErrorContinue()
      - See the upcoming discussion of onErrorStop() for a solution

onErrorStop

```
public final Flux<T> onErrorStop()
```

If an `onErrorContinue(BiConsumer)` variant has been used downstream, reverts to the default 'STOP' mode where errors are terminal events upstream. It can be used for easier scoping of the on next failure strategy or to override the inherited strategy in a sub-stream (for example in a flatMap). It has no effect if `onErrorContinue(BiConsumer)` has not been used downstream.

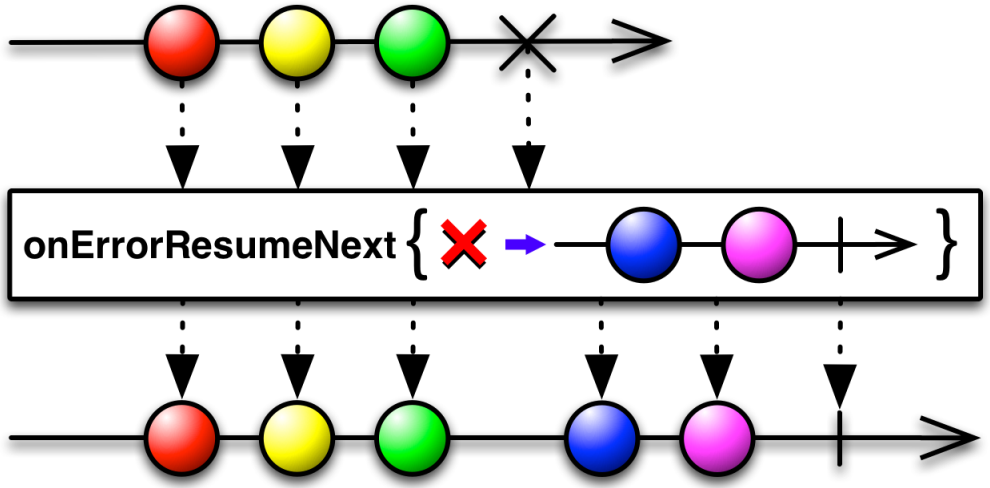
Returns:

a `Flux` that terminates on errors, even if `onErrorContinue(BiConsumer)` was used downstream

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

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
  - RxJava's operator Observable .onErrorResumeNext() works the same



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

Key Error Handling Operators in the Flux Class

- The onErrorResume() operator
  - Subscribe to a returned fallback publisher when any error occurs
  - This operator "swallows" the exception so it won't propagate up the call chain/stack further
  - RxJava's operator Observable .onErrorResumeNext() works the same
  - The Java CompletableFuture exceptionally() method is similar

exceptionally

```
CompletionStage<T> exceptionally(
Function<Throwable,? extends T> fn)
```

Returns a new CompletionStage that, when this stage completes exceptionally, is executed with this stage's exception as the argument to the supplied function. Otherwise, if this stage completes normally, then the returned stage also completes normally with the same value.

Parameters:

`fn` - the function to use to compute the value of the returned CompletionStage if this CompletionStage completed exceptionally

Returns:

the new CompletionStage

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html#exceptionally

Key Error Handling Operators in the Flux Class

- The onErrorStop() operator
  - If an onErrorContinue() variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream

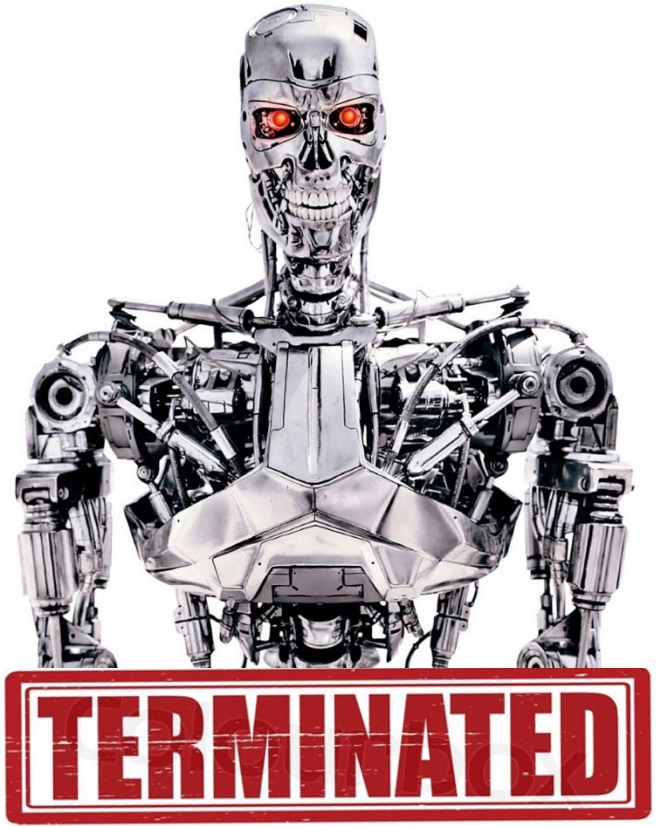
```
Flux<T> onErrorStop()
```

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

Key Error Handling Operators in the Flux Class

- The onErrorStop() operator
  - If an onErrorContinue() variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
    - Returns a Flux that terminates on errors, even if onErrorContinue() was used downstream

```
Flux<T> onErrorStop()
```



Key Error Handling Operators in the Flux Class

- The onErrorStop() operator
  - If an onErrorContinue() variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
  - It can be used for easier scoping of the onNext() failure strategy or to override the inherited strategy in a sub-stream

```
return Flux
   .fromIterable(denominators)

   .map(denominator -> BigFraction
        .valueOf(...,
                 denominator))

   .onErrorResume(__ ->
                   Flux.empty())

   .onErrorStop()

   .collectList()
   ...
```

Prevent a downstream onErrorContinue() from interfering with onErrorResume()

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

Key Error Handling Operators in the Flux Class

- The onErrorStop() operator
  - If an onErrorContinue() variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
  - It can be used for easier scoping of the onNext() failure strategy or to override the inherited strategy in a sub-stream

If onErrorContinue() has not been used downstream onErrorStop() has no effect

onErrorStop

```
public final Flux<T> onErrorStop()
```

If an `onErrorContinue(BiConsumer)` variant has been used downstream, reverts to the default 'STOP' mode where errors are terminal events upstream. It can be used for easier scoping of the on next failure strategy or to override the inherited strategy in a sub-stream (for example in a flatMap). It has no effect if `onErrorContinue(BiConsumer)` has not been used downstream.

Returns:

a `Flux` that terminates on errors, even if `onErrorContinue(BiConsumer)` was used downstream

Key Error Handling Operators in the Flux Class

- The onErrorStop() operator
  - If an onErrorContinue() variant is used downstream, revert to the default 'STOP' mode where errors are terminal events upstream
  - It can be used for easier scoping of the onNext() failure strategy or to override the inherited strategy in a sub-stream
  - RxJava has no direct equivalent
    - Its error handling operators aren't as complicated as Project Reactor's!

End of Key Error Handling Operators in the Flux Class