Key Factory Method Operators in the Flux Class (Part 5)

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
  - Concurrency & scheduler operators
  - Factory method operators
    - These operators create Flux streams in various ways in various Scheduler contexts
      - i.e., the two param version of create()

See en.wikipedia.org/wiki/Factory_method_pattern

Key Factory Method Operators in the Flux Class

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow

```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter, FluxSink
    .OverflowStrategy
    backpressure)
```

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

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
    - Param 1 emits any # of next() signals followed by zero or one error() or complete() signals

```
static <T> Flux<T> create
  (Consumer<? super FluxSink<T>>
   emitter, FluxSink
   .OverflowStrategy
   backpressure)
```

Interface FluxSink<T>

Type Parameters:

T - the value type

```
public interface FluxSink<T>
```

Wrapper API around a downstream Subscriber for emitting any number of next signals followed by zero or one onError/onComplete.

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

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
    - Param 1 emits any # of next() signals followed by zero or one error() or complete() signals
      - Supports more dynamic use cases than the Flux just() & fromIterable() operators

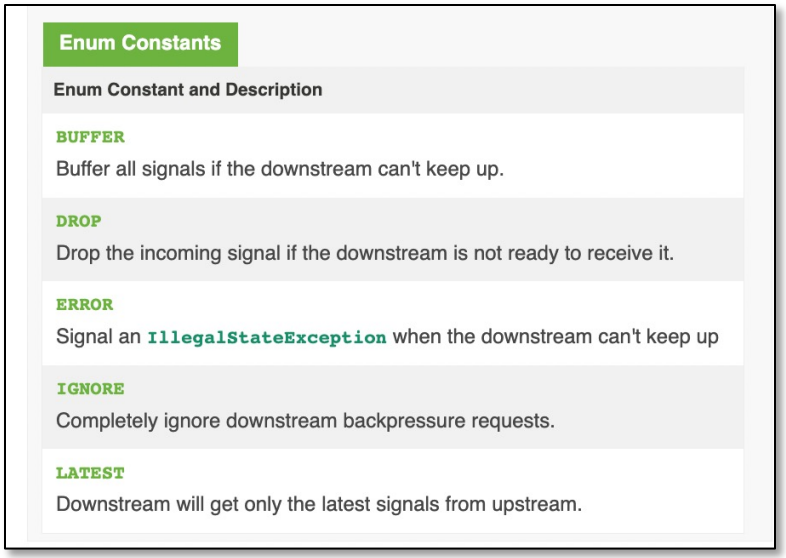
```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter, FluxSink
    .OverflowStrategy
    backpressure)
```

See earlier lesson on "*Key Factory Method Operators in the Flux Class (Part 1)*"

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
    - Param 1 emits any # of next() signals followed by zero or one error() or complete() signals
    - Param 2 defines strategies for handling overflow

```
static <T> Flux<T> create
  (Consumer<? super FluxSink<T>>
   emitter, FluxSink
   .OverflowStrategy
   backpressure)
```



Enum Constants

Enum Constant and Description
BUFFER Buffer all signals if the downstream can't keep up.
DROP Drop the incoming signal if the downstream is not ready to receive it.
ERROR Signal an `IllegalStateException` when the downstream can't keep up
IGNORE Completely ignore downstream backpressure requests.
LATEST Downstream will get only the latest signals from upstream.

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

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
    - Param 1 emits any # of next() signals followed by zero or one error() or complete() signals
    - Param 2 defines strategies for handling overflow

```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter, FluxSink
    .OverflowStrategy
    backpressure)
```

"backpressure" is an odd choice of terms here

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

Key Factory Method Operators in the Flux Class

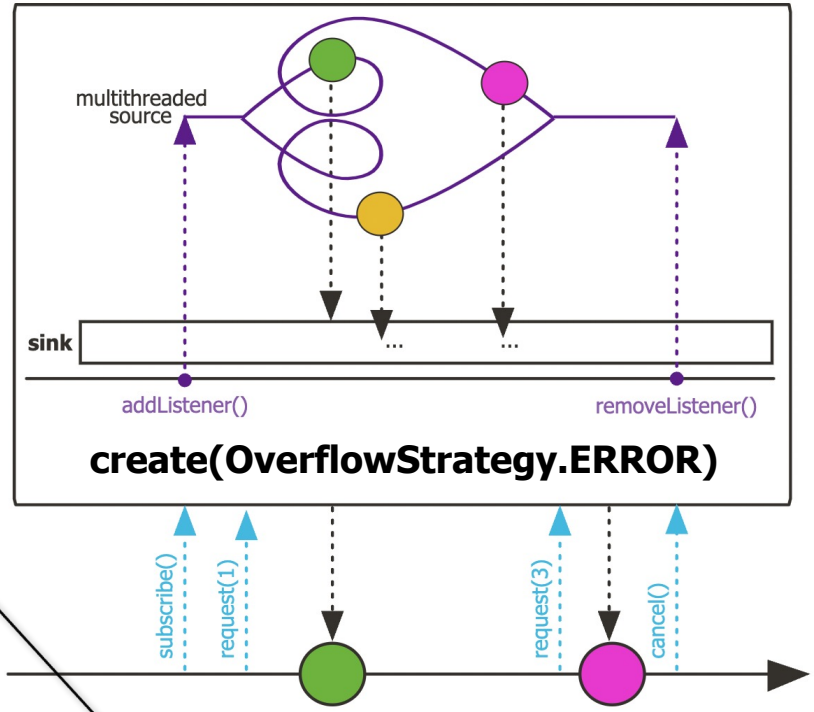
- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
    - Param 1 emits any # of next() signals followed by zero or one error() or complete() signals
    - Param 2 defines strategies for handling overflow
    - Returns a Flux that emits all the elements generated by the FluxSink

```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter, FluxSink
    .OverflowStrategy
    backpressure)
```

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow

```
Flux
   .create(makeEmitter(count,
                       sb),
           FluxSink
           .OverflowStrategy
           .ERROR)
   ...
   .subscribe
      (blockingSubscriber);
```



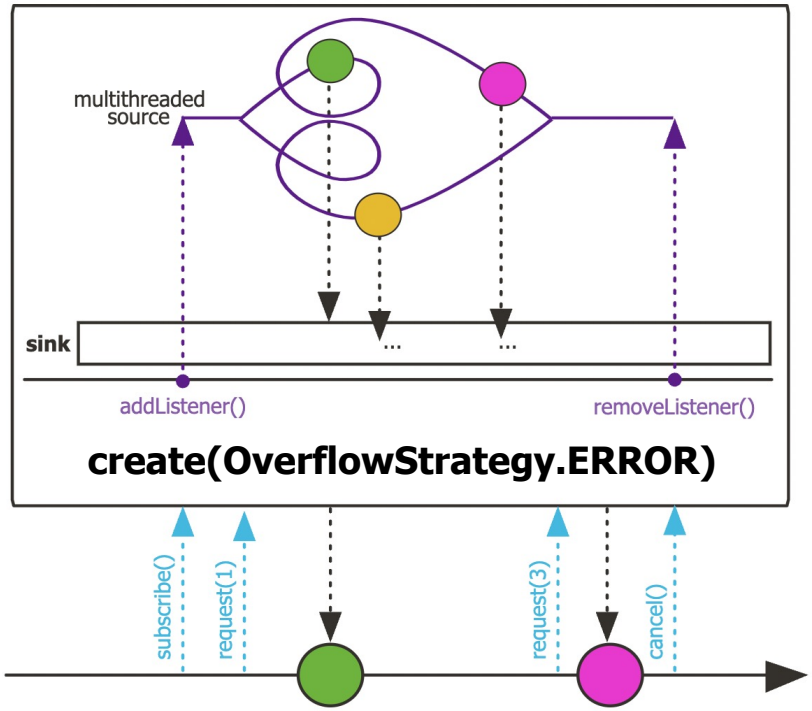
Rapidly emit a stream of random BigFraction objects in one fell swoop

See github.com/douglascraigschmidt/LiveLessons/tree/master/Reactive/flux/ex4

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow

```
Flux
   .create(makeEmitter(count,
                       sb),
           FluxSink
           .OverflowStrategy
           .ERROR)
   ...
   .subscribe
       (blockingSubscriber);
```

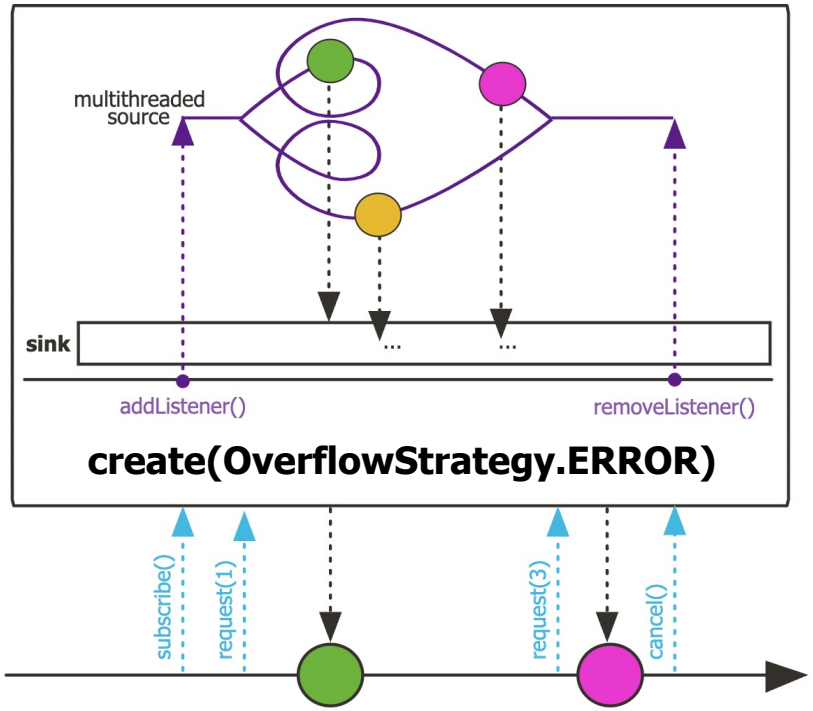


Throw exception when events can't be processed immediately

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow

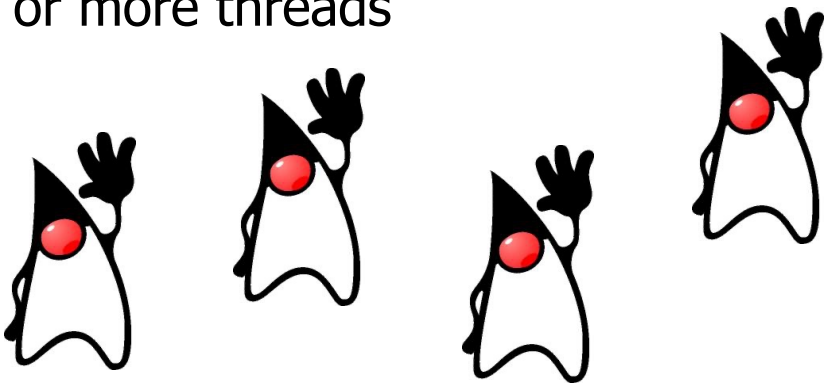
```
Flux
   .create(makeEmitter(count,
                       sb),
           FluxSink
           .OverflowStrategy
           .ERROR)
   ...
   .subscribe
       (blockingSubscriber);
```

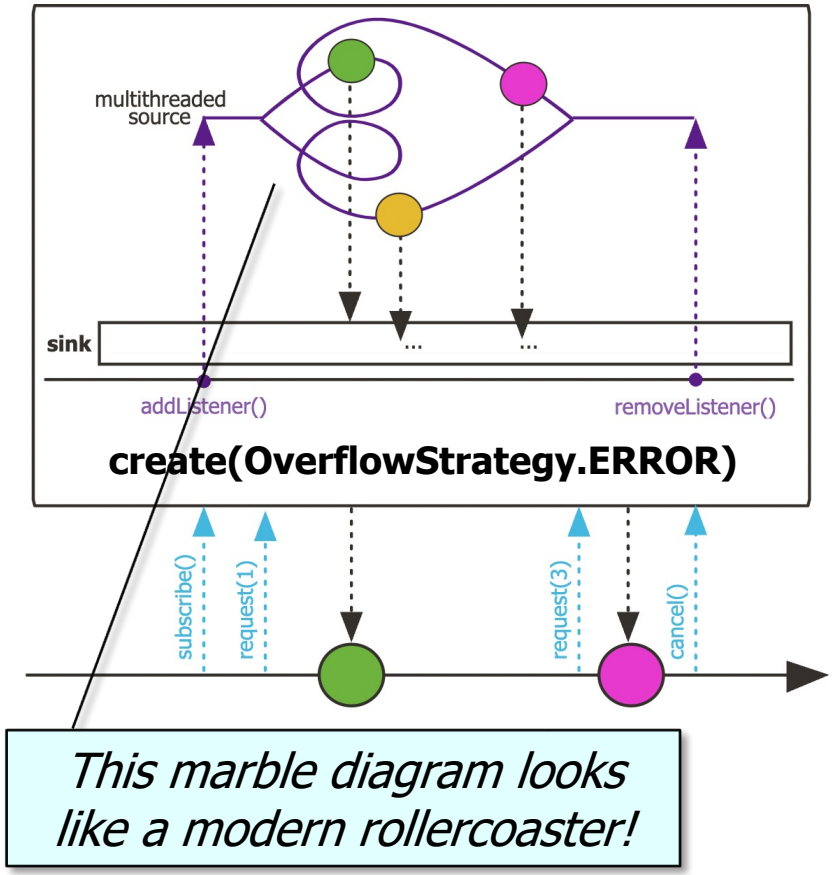


Start all the async processing

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
  - Elements can be emitted from one or more threads





Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
  - Elements can be emitted from one or more threads
  - RxJava's Flowable.create() is similar

create

```
@CheckReturnValue
 @NonNull
 @BackpressureSupport(value=SPECIAL)
 @SchedulerSupport(value="none")
public static <T> @NonNull Flowable<T> create(@NonNull @NonNull FlowableOnSubscribe<T> source,
        @NonNull @NonNull BackpressureStrategy mode)
```

Provides an API (via a cold `Flowable`) that bridges the reactive world with the callback-style, generally non-backpressured world.

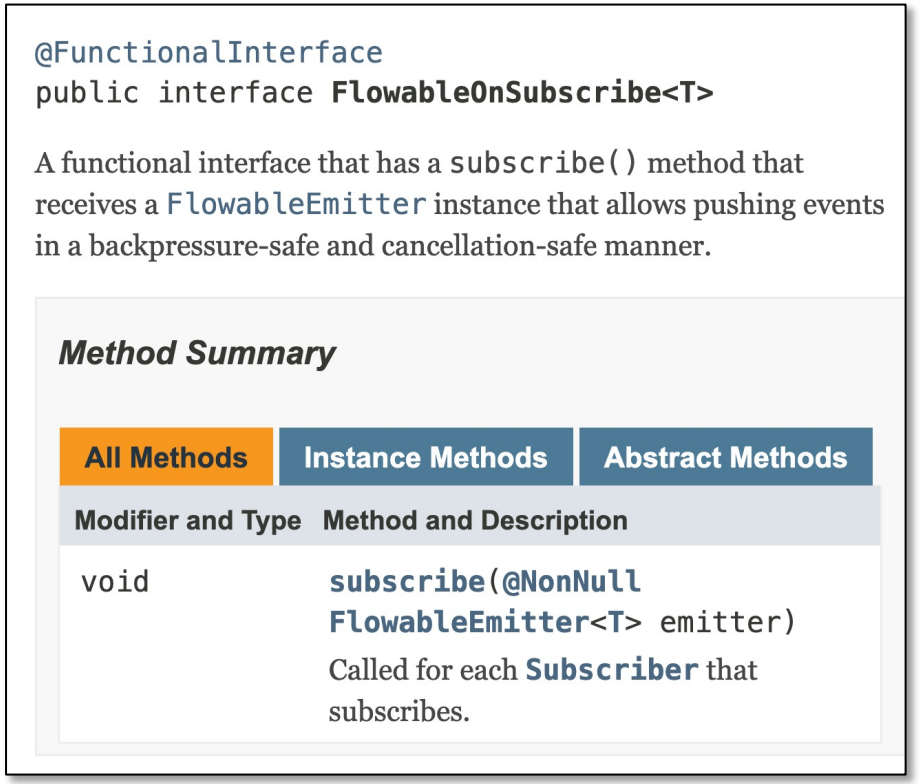
Example:

```
 Flowable.<Event>create(emitter -> {
     Callback listener = new Callback() {
         @Override
         public void onEvent(Event e) {
             emitter.onNext(e);
             if (e.isLast()) {
                 emitter.onComplete();
             }
         }

         @Override
         public void onFailure(Exception e) {
             emitter.onError(e);
         }
     };

     AutoCloseable c = api.someMethod(listener);

     emitter.setCancellable(c::close);

 }, BackpressureStrategy.BUFFER);
```

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

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
  - Elements can be emitted from one or more threads
  - RxJava's Flowable.create() is similar
    - However, the data types passed to create() differ
      - i.e., FlowableOnSubscribe vs. Consumer<FluxSync>



```
@FunctionalInterface
public interface FlowableOnSubscribe<T>
```

A functional interface that has a `subscribe()` method that receives a `FlowableEmitter` instance that allows pushing events in a backpressure-safe and cancellation-safe manner.

Method Summary

All Methods	Instance Methods	Abstract Methods

Modifier and Type	Method and Description
`void`	**`subscribe`**`(@NonNull` **`FlowableEmitter`**`<T> emitter)` Called for each **`Subscriber`** that subscribes.

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

Key Factory Method Operators in the Flux Class

- The two param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously & that handles overflow
  - Elements can be emitted from one or more threads
  - RxJava's Flowable.create() is similar
- Java Streams generate() does not (need to) handle backpressure

generate

```
static <T> Stream<T> generate(Supplier<T> s)
```

Returns an infinite sequential unordered stream where each element is generated by the provided `Supplier`. This is suitable for generating constant streams, streams of random elements, etc.

Type Parameters:

`T - the type of stream elements`

Parameters:

`s - the Supplier of generated elements`

Returns:

`a new infinite sequential unordered Stream`

Generate a stream of random, large, & unreduced big fractions

```
Stream
    .generate(() -> BigFractionUtils
        .makeBigFraction(new Random(),
                         false))
```

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

End of Key Factory Method Operators in the Flux Class (Part 5)