Key Factory Method Operators in the Flux 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 Flux operators
  - Concurrency operators
  - Scheduler operators
  - Factory method operators
    - These operators create Flux streams in various ways in various Scheduler contexts
      - i.e., the one-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 one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously

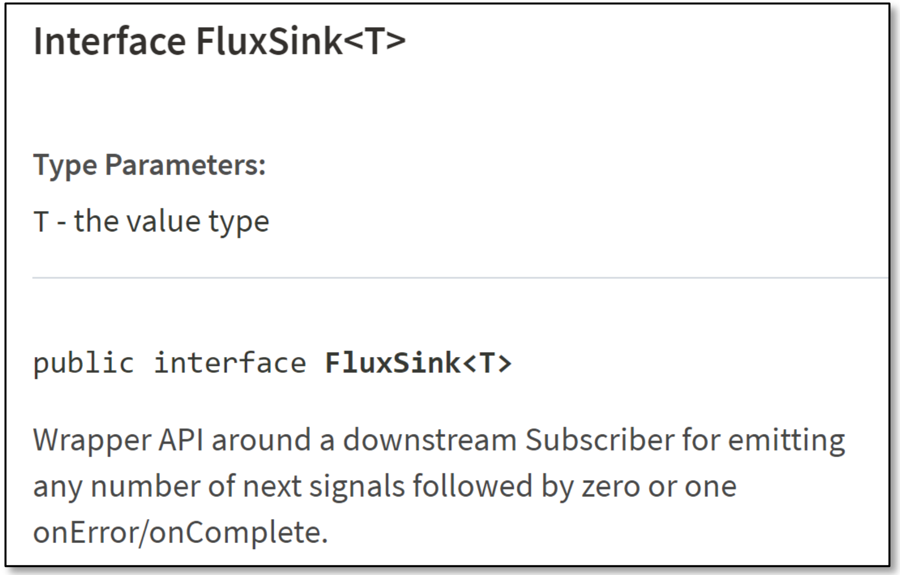
```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter)
```

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

Key Factory Method Operators in the Flux Class

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

```
static <T> Flux<T> create
   (Consumer<? super FluxSink<T>>
    emitter)
```



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 one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
    - The param 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)
```

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

Key Factory Method Operators in the Flux Class

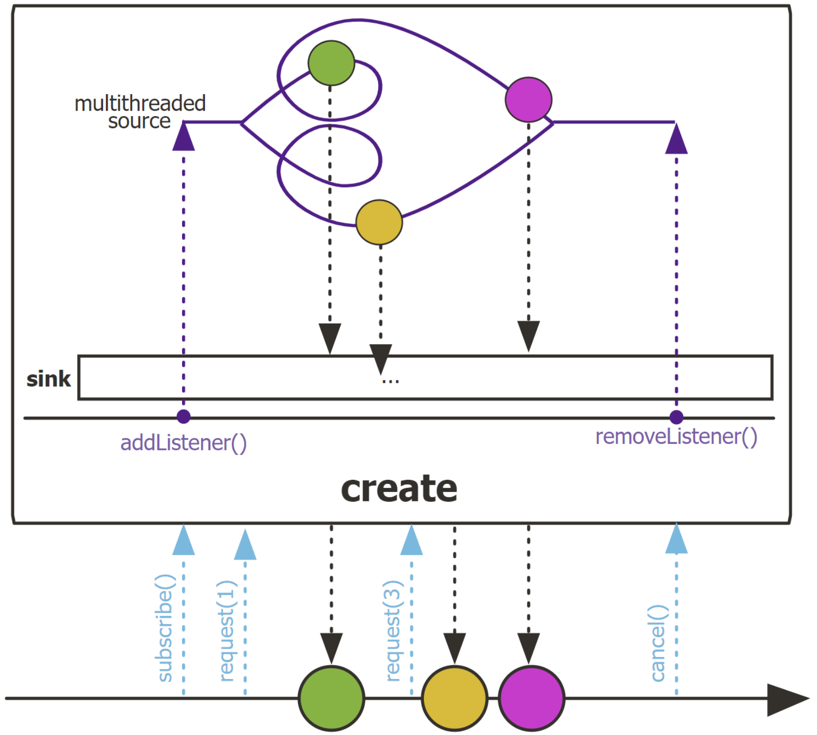
- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
    - The param emits any # of next() signals followed by zero or one error() or complete() signals
    - Returns a Flux that emits all the elements generated by the FluxSink

```
static <T> Flux<T> create
    (Consumer<? super FluxSink<T>>
     emitter)
```

Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously

```
static <T> Flux<T> generate
             (Supplier<T> supplier,
              long count) {
  return Flux.create(sink -> {
    for(int i = 0; i < count; ++i)
      sink.next(supplier.get());

    sink.complete();
  });
}
```



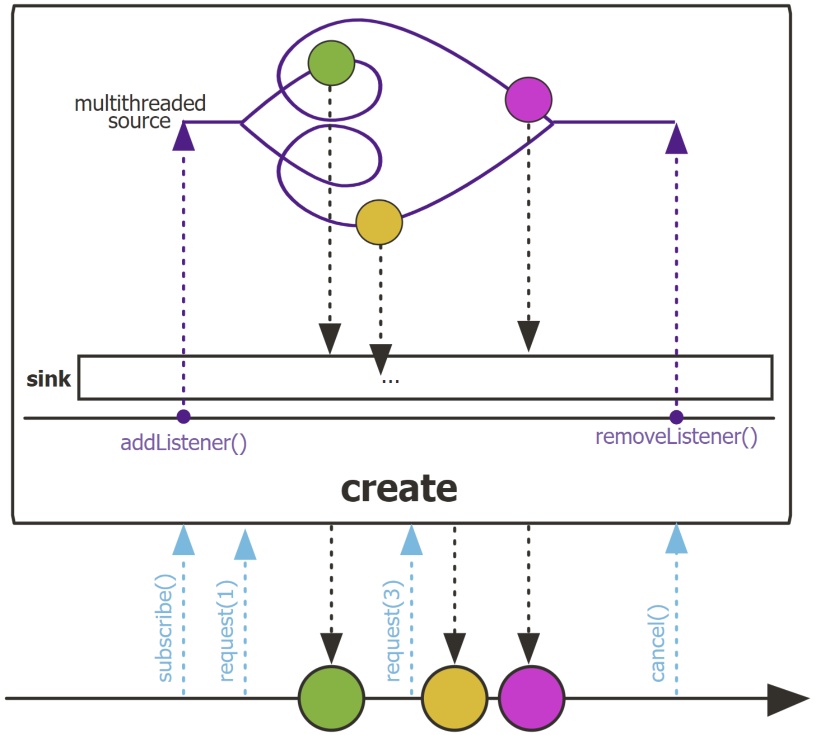
Synchronously generate 'count' instances of what's returned by supplier.get()

See [Reactive/Flux/ex1/src/main/java/utils/ReactorUtils.java]()

Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously

```
static <T> Flux<T> generate
             (Supplier<T> supplier,
              long count) {
  return Flux.create(sink -> {
    for(int i = 0; i < count; ++i)
      sink.next(supplier.get());

    sink.complete();
  });
}
```

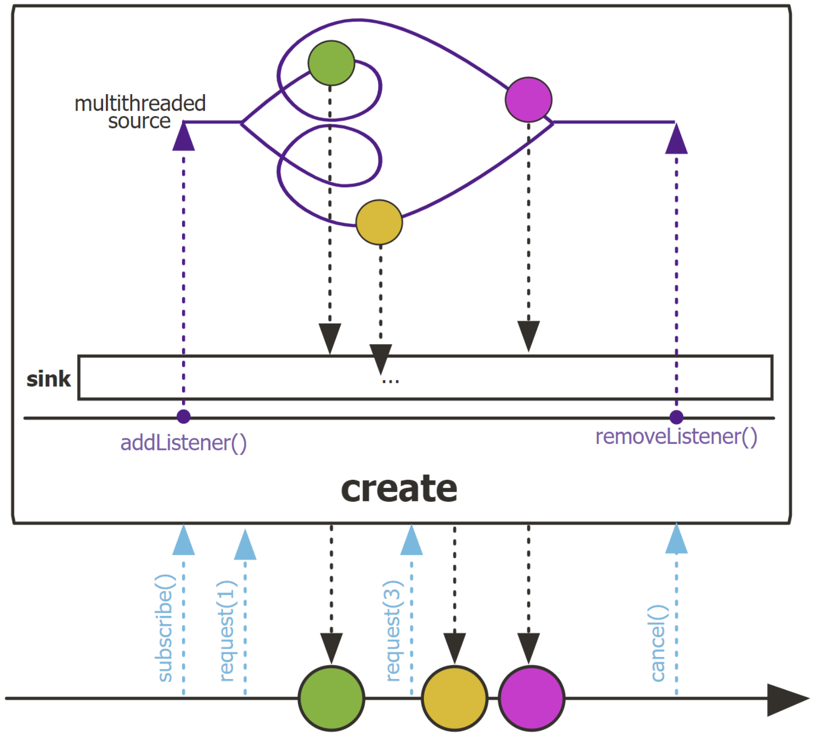


Generate the next element & emit it

Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously

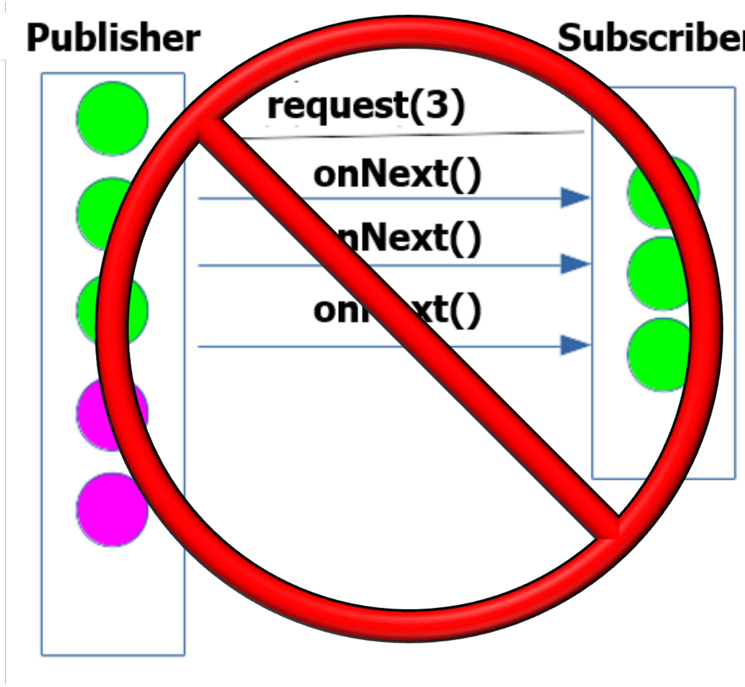
```
static <T> Flux<T> generate
           (Supplier<T> supplier,
            long count) {
  return Flux.create(sink -> {
    for(int i = 0; i < count; ++i)
      sink.next(supplier.get());

    sink.complete();
  });
}
```



Indicate the generator is finished

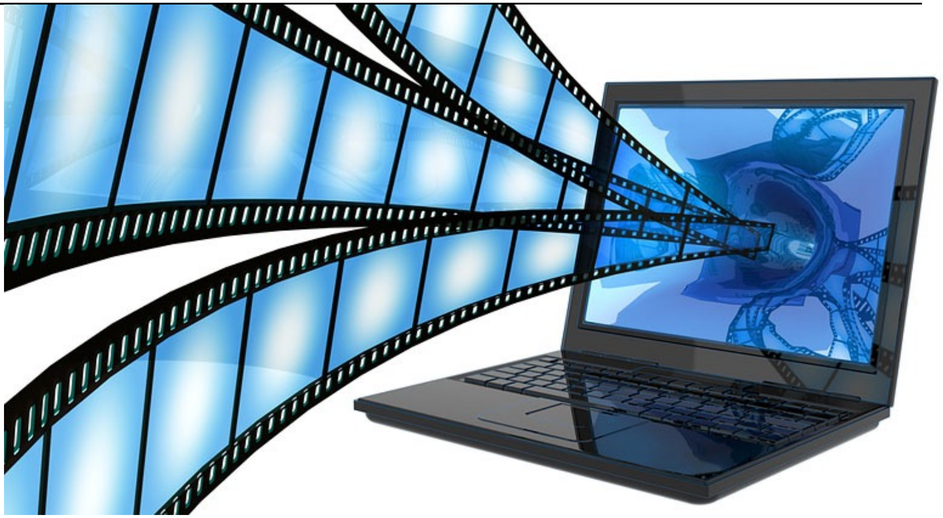
Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure



Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
    - It can thus emit a (potentially endless) stream of elements at a high rate



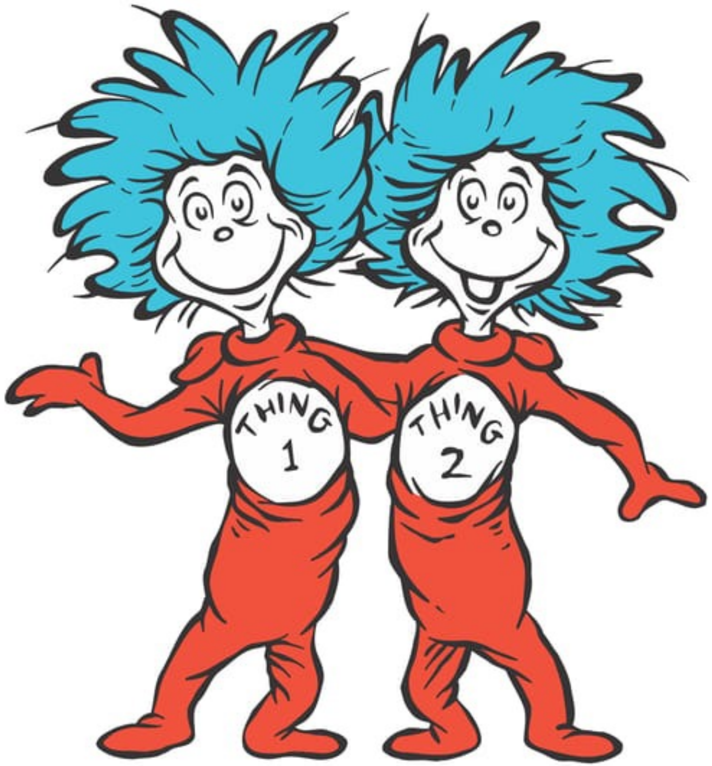
Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
    - It can thus emit a (potentially endless) stream of elements at a high rate
    - A fast publisher can overwhelm memory/processing resources of a slower consumer

See www.wideopeneats.com/i-love-lucy-chocolate-factory

Key Factory Method Operators in the Flux Class

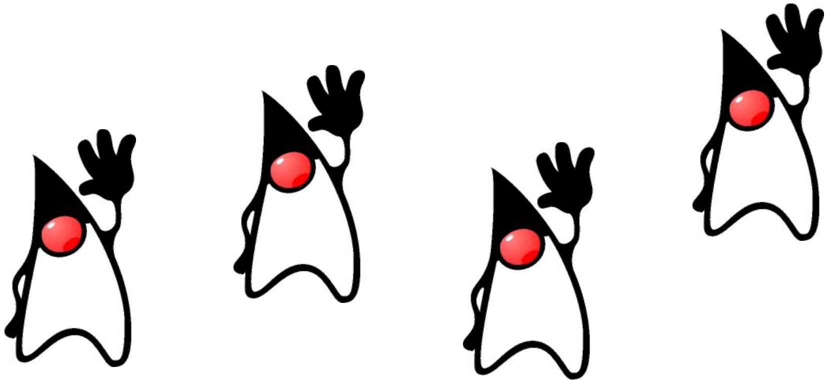
- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
    - It can thus emit a (potentially endless) stream of elements at a high rate
    - A fast publisher can overwhelm memory/processing resources of a slower consumer

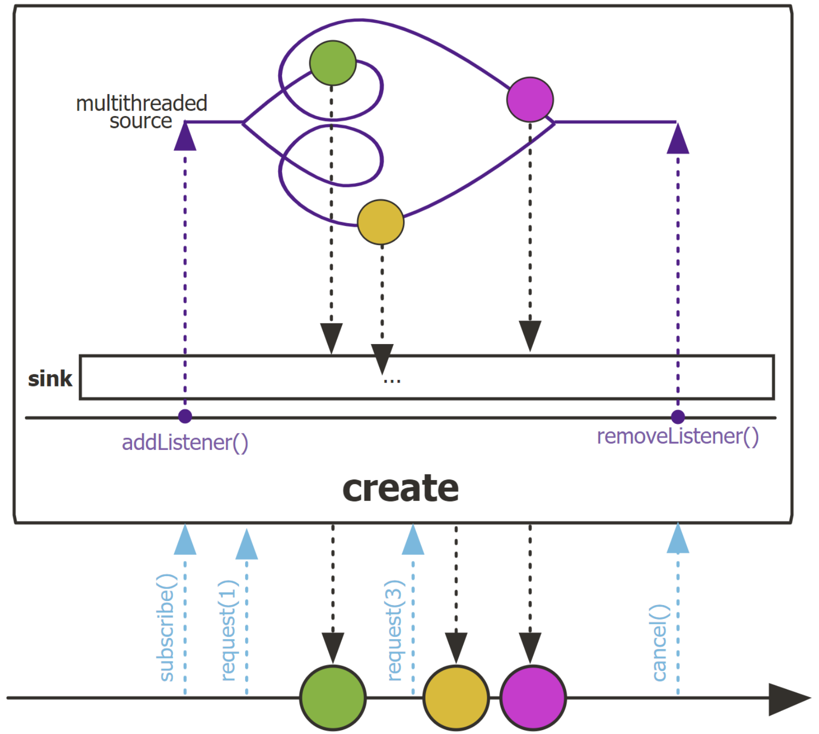


Fortunately, Project Reactor Flux provides two solutions we'll discuss shortly!

Key Factory Method Operators in the Flux Class

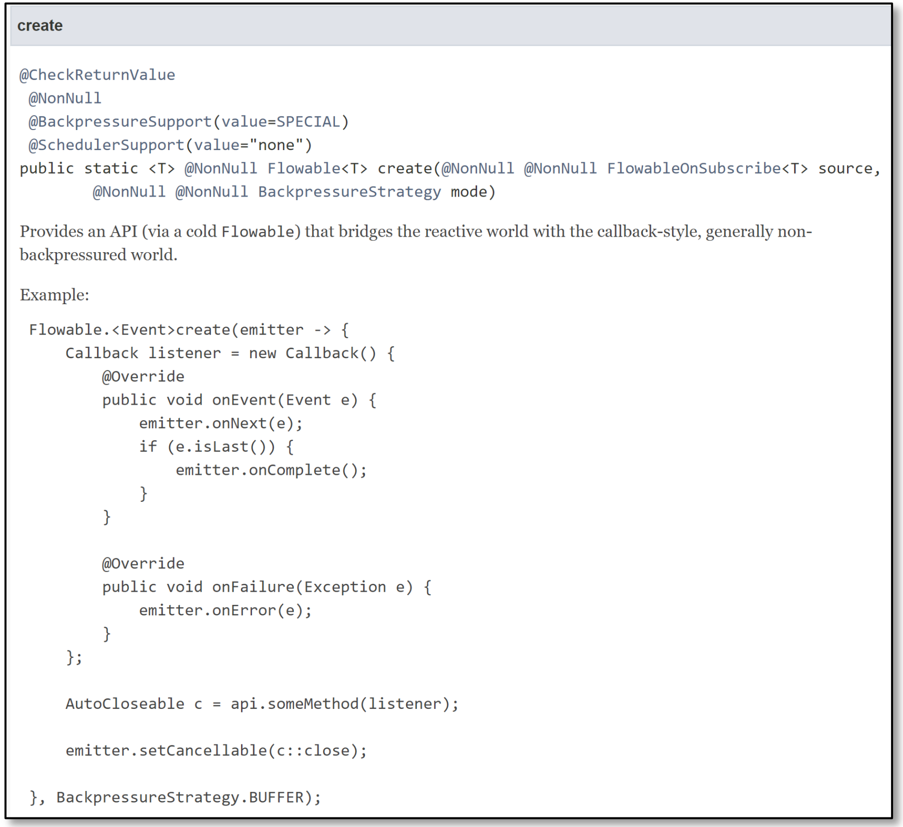
- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
  - Elements can be emitted from one or more threads





Key Factory Method Operators in the Flux Class

- The one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
  - 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>



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.

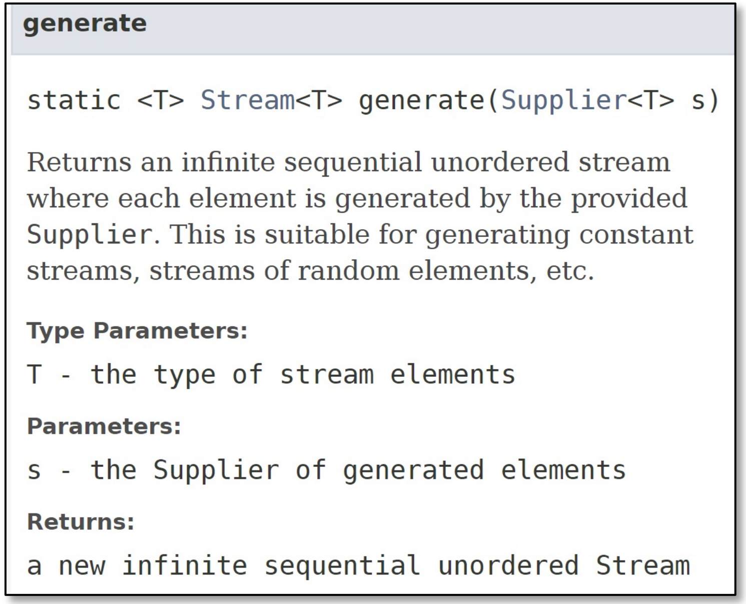
Example:

```
 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 one param create() operator
  - Create a Flux capable of emitting multiple elements synchronously or asynchronously
  - Does not support backpressure
  - Elements can be emitted from one or more threads
  - RxJava's Flowable.create() is similar
  - Similar to the generate() method in Java Streams



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`

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

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

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 2)