Key Factory Method Operators in the Observable Class (Part 3)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

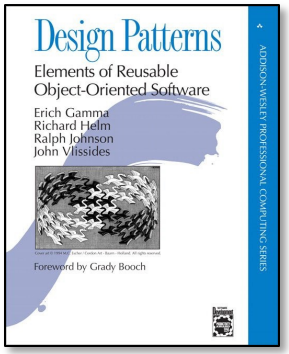
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Recognize key Observable operators
  - Factory method operators
    - These operators create Observable streams in various ways
      - e.g., generate()



See en.wikipedia.org/wiki/Factory_method_pattern

Key Factory Method Operators in the Observable Class

Key Factory Method Operators in the Observable Class

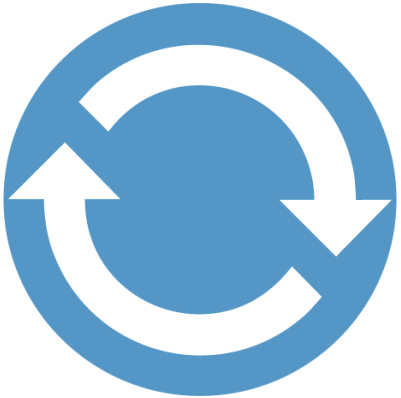
- The generate() operator
  - Returns a synchronous & stateless generator of values

```
static <T> Observable<T>
generate
   (Callable<Emitter<T> generator)
```

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

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
    - The generator param is called in a loop when a downstream Observer subscribes



```
static <T> Observable<T>
generate
   (Callable<Emitter<T>> generator)
```

`public interface Emitter<T>`

Base interface for emitting signals in a push-fashion in various generator-like source operators (create, generate).

Note that the `onNext(T)`, `onError(java.lang.Throwable)` and `onComplete()` methods provided to the function via the `Emitter` instance should be called synchronously, never concurrently. Calling them from multiple threads is not supported and leads to an undefined behavior.

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	`onComplete()` Signal a completion.
void	`onError(@NonNull Throwable error)` Signal a `Throwable` exception.
void	`onNext(T value)` Signal a normal value.

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

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
    - The generator param is called in a loop when a downstream Observer subscribes
      - This callback should call onNext(), onError(), or onComplete() to signal a value or a terminal event

```
static <T> Observable<T>
generate
   (Callable<Emitter<T>> generator)
```

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

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
    - The generator param is called in a loop when a downstream Observer subscribes
    - The new Observable instance is returned

```
static <T> Observable<T>
generate
   (Callable<Emitter<T>> generator)
```

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
    - The generator param is called in a loop when a downstream Observer subscribes
  - The new Observable instance is returned
    - This observable is "cold," which only emits item upon subscription

```
static <T> Observable<T>
generate
   (Callable<Emitter<T>> generator)
```

See medium.com/tompee/rxjava-ninja-hot-and-cold-observables-19b30d6cc2fa

Key Factory Method Operators in the Observable Class

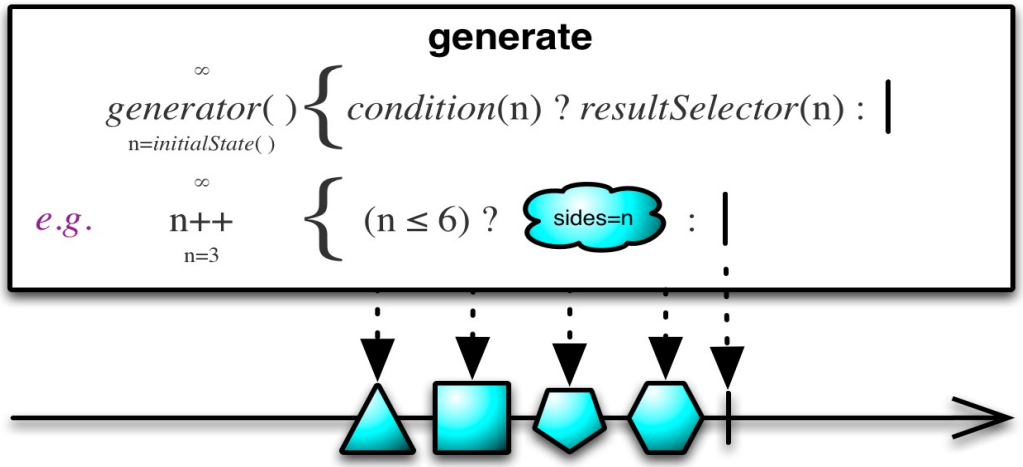
- The generate() operator
  - Returns a synchronous & stateless generator of values
    - The generator param is called in a loop when a downstream Observer subscribes
  - The new Observable instance is returned
    - This observable is "cold," which only emits item upon subscription
    - Each observer will have its own set of items emitted to them

```
static <T> Observable<T>
generate
   (Callable<Emitter<T>> generator)
```

See medium.com/tompee/rxjava-ninja-hot-and-cold-observables-19b30d6cc2fa

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure

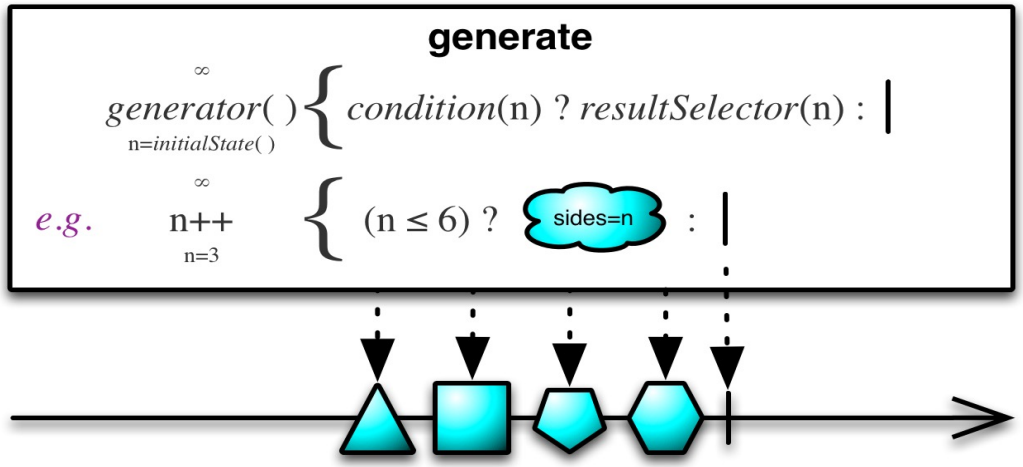



```
Observable
   .generate((Emitter<BigFraction> emit) -> emit
             .onNext(BigFractionUtils
                     .makeBigFraction(sRANDOM, false)))
   ...
```

See www.baeldung.com/rxjava-backpressure

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure



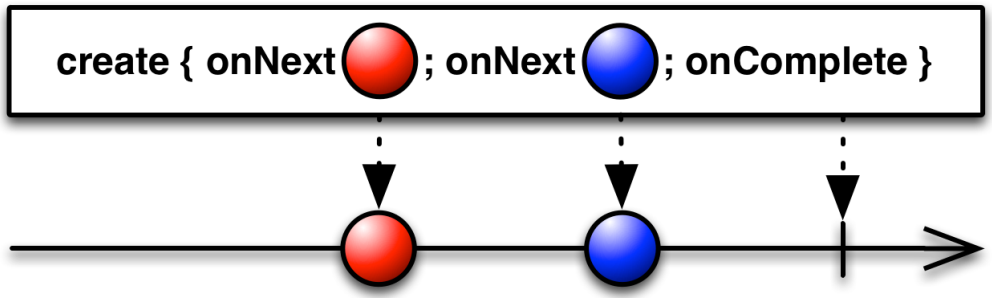

```
Observable
   .generate((Emitter<BigFraction> emit) -> emit
                .onNext(BigFractionUtils
                           .makeBigFraction(sRANDOM, false)))
   ...
```

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

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

Key Factory Method Operators in the Observable Class

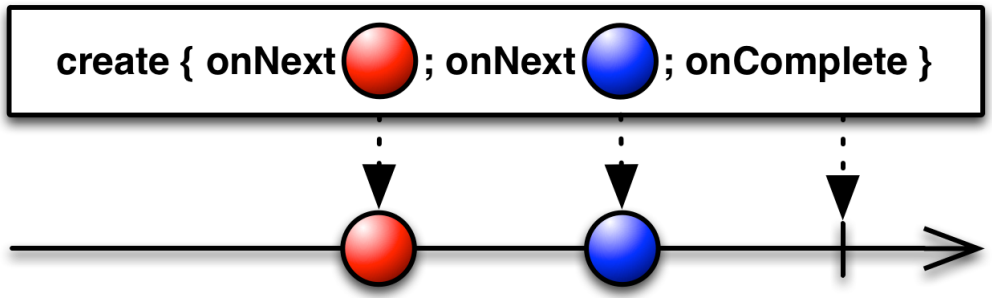
- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure
    - In contrast, create() can produce events whenever it wishes to do so



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

Key Factory Method Operators in the Observable Class

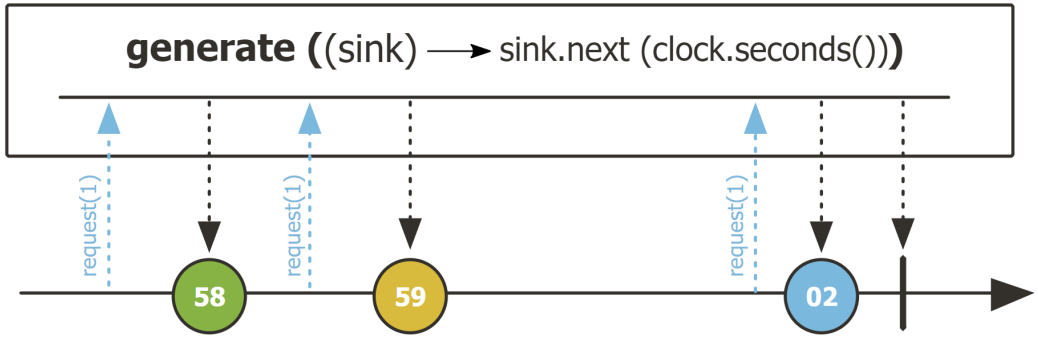
- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure
    - In contrast, create() can produce events whenever it wishes to do so
      - i.e., it ignores backpressure



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

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure
- Project Reactor's Flux.generate() operator works in a similar way




```
Flux.generate
   ((SynchronousSink<BigFraction>
     sink) -> sink
    .next(BigFractionUtils
          .makeBigFraction(sRANDOM,
                           false)))
...
```

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

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

Key Factory Method Operators in the Observable Class

- The generate() operator
  - Returns a synchronous & stateless generator of values
  - It can only generate one event at a time, which supports a simple form of backpressure
  - Project Reactor's Flux.generate() operator works the same
  - Similar to the Stream.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 Observable Class (Part 3)