Key Action Operators in the Flux Class (Part 1)

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
  - Action operators
    - These operators don’t modify a Flux, but instead use it for side effects
      - e.g., doOnNext() & doOnError()

Key Action Operators in the Flux Class

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item

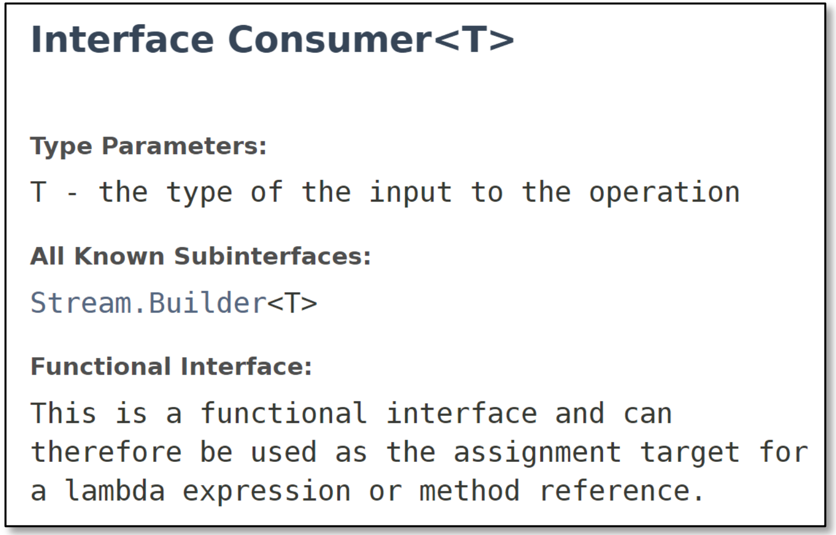
```
Flux<T> doOnNext
  (Consumer<? super T> onNext)
```

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

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
    - The behavior is passed as a Consumer param that's called on successful completion

```
Flux<T> doOnNext
  (Consumer<? super T> onNext)
```



Interface Consumer<T>

Type Parameters:

T - the type of the input to the operation

All Known Subinterfaces:

Stream.Builder<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/Consumer.html

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
    - The behavior is passed as a Consumer param that's called on successful completion

```
Flux<T> doOnNext
  (Consumer<? super T> onNext)
```

i.e., it is a "callback," which is designed to have "side-effects"

See en.wikipedia.org/wiki/Callback_(computer_programming)

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
    - The behavior is passed as a Consumer param that's called on successful completion
      - doOnNext() is skipped if an unhandled error (exception) occurs in the stream

```
Flux<T> doOnNext
  (Consumer<? super T> onNext)
```

See upcoming discussion of the doOnError() operator

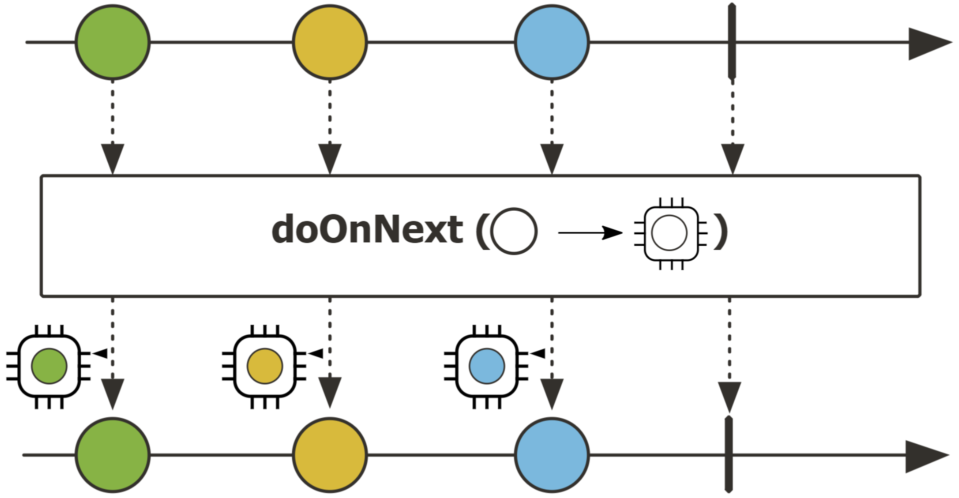
Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
    - The behavior is passed as a Consumer param that's called on successful completion
    - Returns a Flux that is not modified at all
      - i.e., the type and/or value of its elements are not changed

```
Flux<T> doOnNext
  (Consumer<? super T> onNext)
```

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
  - Used primarily for getting visibility into a Flux stream
    - e.g., debugging or logging



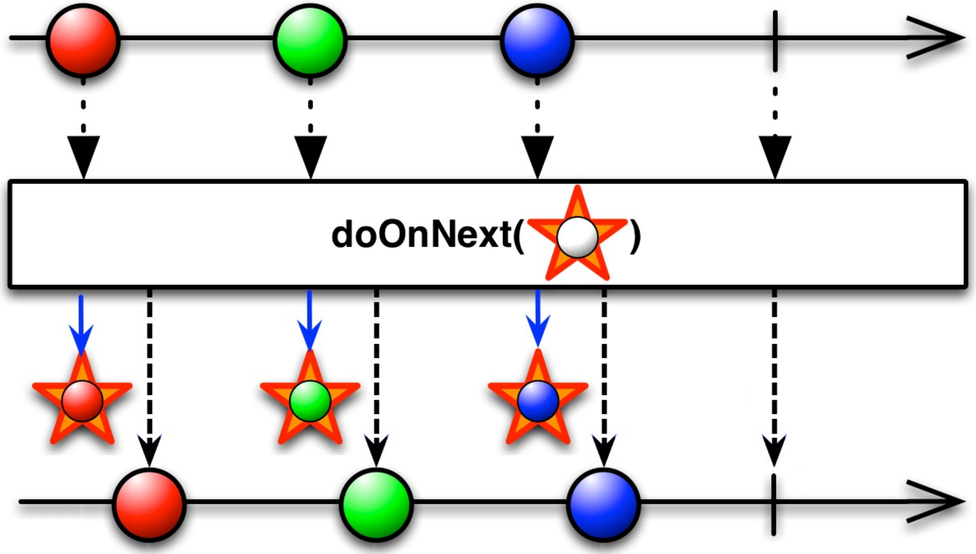
Log each BigFraction value on success (otherwise skip)

```
Flux
  .fromIterable(bigFractionList)
  .doOnNext(bf ->
            logBigFraction(sUnreducedFraction, bf, sb))
  ...
```

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
  - Used primarily for getting visibility into a Flux stream
  - RxJava's Observable.doOnNext() works the same



```
Observable
  .fromIterable(bigFractionList)
  .doOnNext(bf ->
            logBigFraction(sUnreducedFraction, bf, sb))
  ...
```

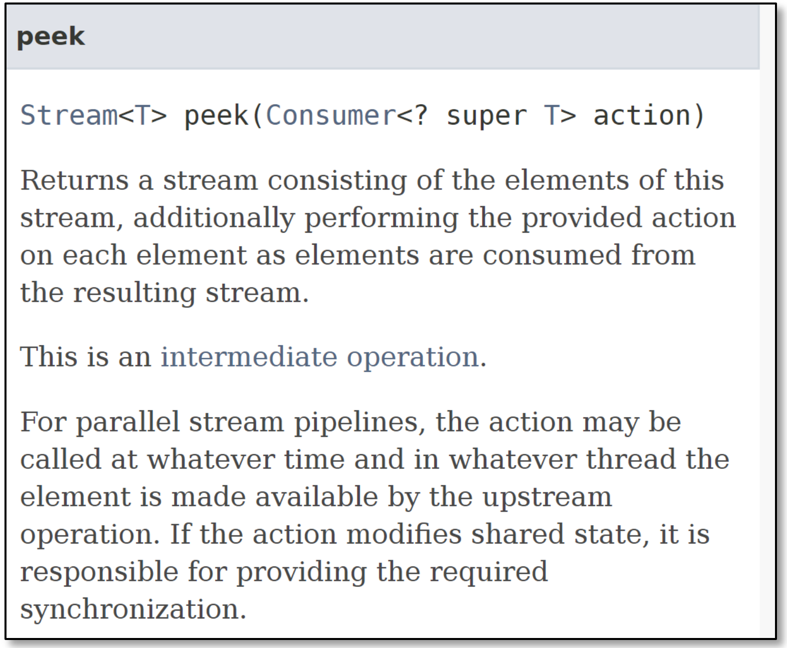
Log each BigFraction value on success (otherwise skip)

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

Key Action Operators in the Flux Class

- The doOnNext() operator
  - Add a behavior triggered when a Flux emits an item
  - Used primarily for getting visibility into a Flux stream
  - RxJava's Observable.doOnNext() works the same
  - Similar to Stream.peek() in Java Streams

```
List<String> collect = List
   .of("a", "b", "c").stream()
   .peek(System.out::println)
   .map(String::toUpperCase).toList();
```



peek

`Stream<T> peek(Consumer<? super T> action)`

Returns a stream consisting of the elements of this stream, additionally performing the provided action on each element as elements are consumed from the resulting stream.

This is an intermediate operation.

For parallel stream pipelines, the action may be called at whatever time and in whatever thread the element is made available by the upstream operation. If the action modifies shared state, it is responsible for providing the required synchronization.

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

Key Action Operators in the Flux Class

- The doOnError() operator
  - Add a behavior triggered when a Flux completes with an error

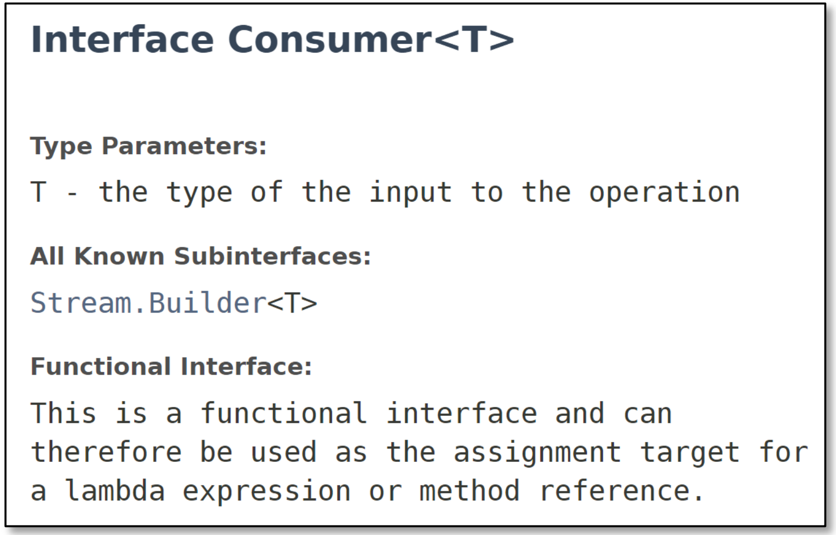
```
Flux<T> doOnError
  (Consumer<? super Throwable>
   onError)
```

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

Key Action Operators in the Flux Class

- The doOnError() operator
  - Add a behavior triggered when a Flux completes with an error
    - The Consumer param designates the behavior called on unsuccessful completion

```
Flux<T> doOnError
  (Consumer<? super Throwable>
   onError)
```



Interface Consumer<T>

Type Parameters:

`T - the type of the input to the operation`

All Known Subinterfaces:

`Stream.Builder<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/Consumer.html

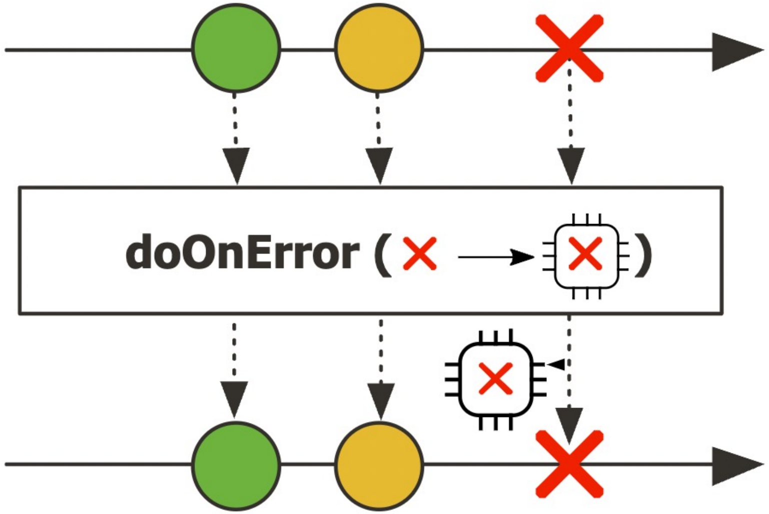
Key Action Operators in the Flux Class

- The doOnError() operator
  - Add a behavior triggered when a Flux completes with an error
    - The Consumer param designates the behavior called on unsuccessful completion
    - Returns a Flux that is not modified at all
      - i.e., the type and/or value of its elements are not changed

```
Flux<T> doOnError
  (Consumer<? super Throwable>
   onError)
```

Key Action Operators in the Flux Class

- The doOnError() operator
  - Add a behavior triggered when a Flux completes with an error
  - Used primarily for getting visibility into a Flux chain
    - e.g., debugging or logging



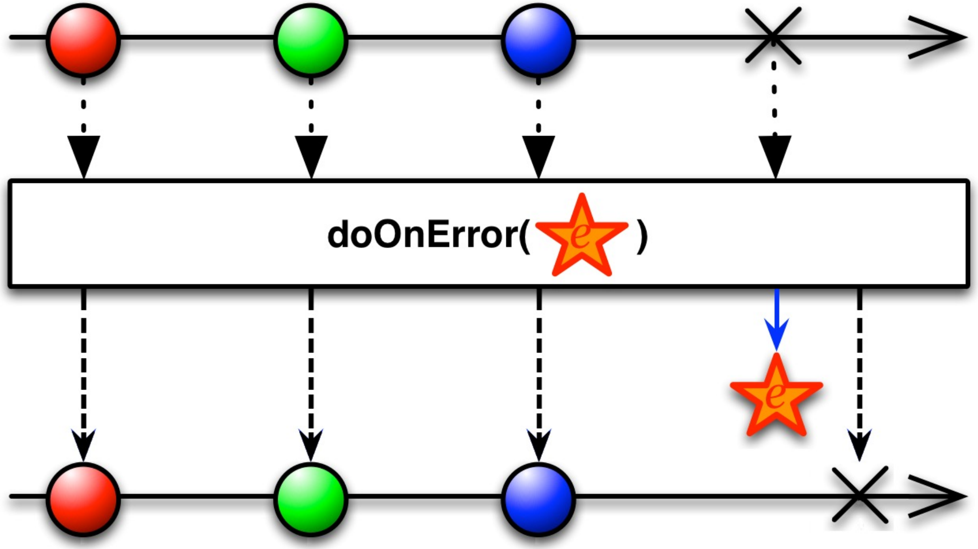
Log each BigFraction value on failure (otherwise skip)

```
Flux
  .fromIterable(bigFractionList)
  .map(bf -> bf.divide(BigFraction.ZERO))
  .doOnError(ex -> logError(ex))
  ...
```

See Reactive/flux/ex1/src/main/java/FluxEx.java

Key Action Operators in the Flux Class

- The doOnError() operator
  - Add a behavior triggered when a Flux completes with an error
  - Used primarily for getting visibility into a Flux chain
  - RxJava's Observable.doOnNext() works the same



```
Observable
   .fromIterable(bigFractionList)
   .map(bf -> bf.divide(BigFraction.ZERO))
   .doOnError(ArithmeticException.class, ex -> logError(ex))
   ...
```

Log each BigFraction value on error (otherwise skip)

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

End of Key Action Operators in the Flux Class (Part 1)