Java Reactive Streams

Reactive Streams

The purpose of Reactive Streams is to provide a standard for asynchronous stream processing with non-blocking backpressure.

The main goal of Reactive Streams is to govern the exchange of stream **data across an asynchronous boundary** – think passing elements on to another thread or thread-pool — while ensuring that the receiving side **is not forced to buffer arbitrary amounts of data**.

Goals, Design and Scope

- Handling streams of data—especially “live” data whose **volume is not predetermined**—requires special care in an asynchronous system.

- The most prominent issue is that resource consumption needs to be carefully controlled such that a fast data source **does not overwhelm the stream destination**.

- Asynchrony is needed in order to enable the **parallel use of computing resources**, on collaborating network hosts or multiple CPU cores

Reactive Streams is a standard for JVM

- process a potentially unbounded number of elements in sequence,
- asynchronously passing elements between components,
- with mandatory non-blocking backpressure.

API Components

- Publisher
- Subscriber
- Subscription
- Processor

A ***Publisher*** is a provider of a potentially unbounded number of sequenced elements, publishing them according to the demand received from its ***Subscribe***r(s)

Publisher code

```
public interface Publisher<T> {
    public void subscribe(Subscriber<? super T> s);
}
```

Subscriber code

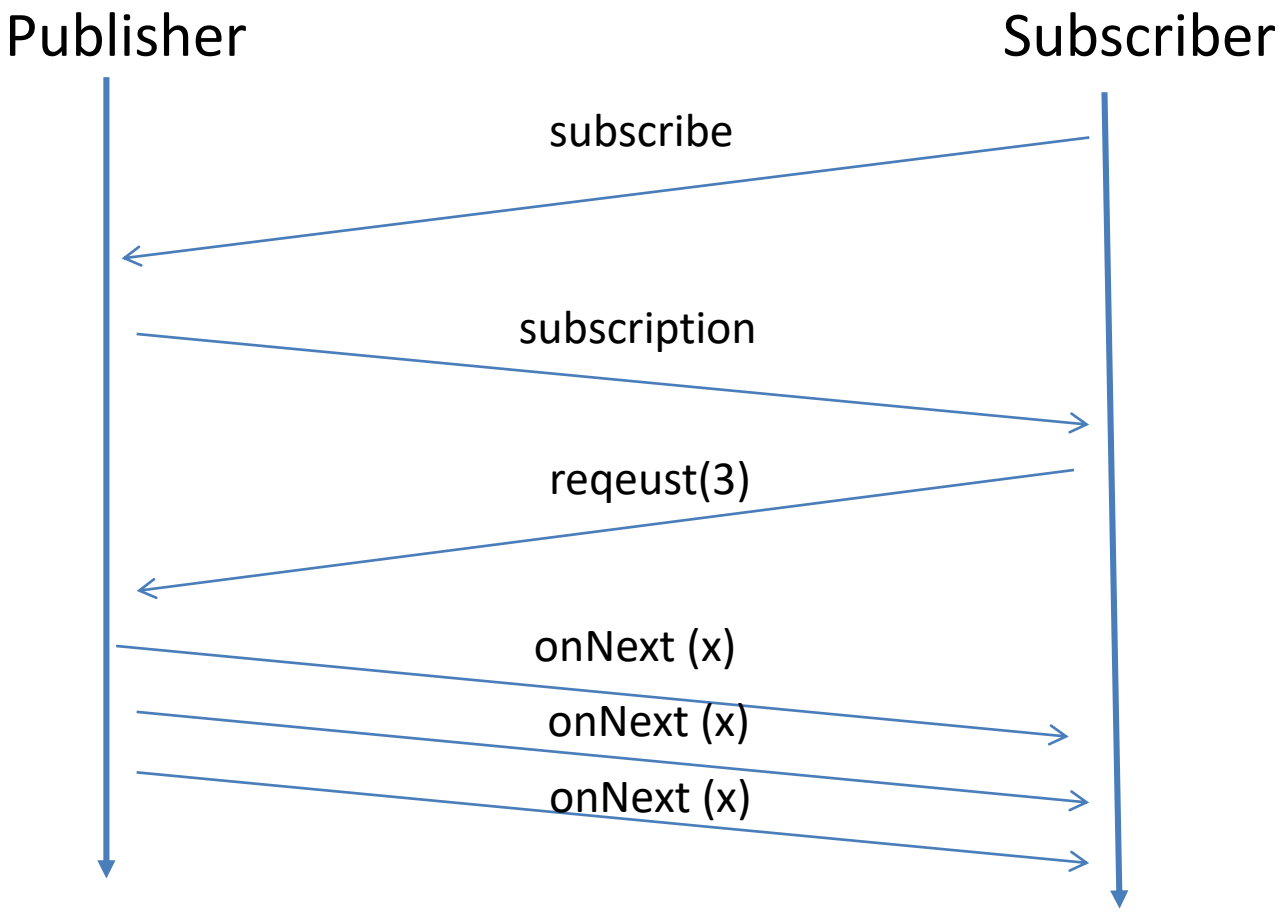
```
public interface Subscriber<T> {
    public void onSubscribe(Subscription s);
    public void onNext(T t);
    public void onError(Throwable t);
    public void onComplete();
}
```

Subscription code

```
public interface Subscription {
    public void request(long n);
    public void cancel();
}
```

A Subscriber MUST signal demand via
Subscription.request(long n)
to receive onNext signals.

Sequence Diagram



Processor code

```
public interface Processor<T, R> extends Subscriber<T>, Publisher<R> {
}
```

A Processor represents a processing stage—which is both a Subscriber and a Publisher and MUST obey the contracts of both.

Legal

This project is a collaboration between engineers from:

- Kaazing,
- Netflix,
- Pivotal,
- RedHat,
- Twitter,
- Typesafe

End