****

Contents

[**I – Sơ lược về Flow API** 1](#_Toc110403277)

[**II – Các thành phần của Flow API** **Flow API** chứa 5 thành phần cơ bản : 2](#_Toc110403278)

[*1. Interface* ***Subscriber****<T> có bốn phương thức:* 2](#_Toc110403279)

[*2.Interface* ***Subscription*** *có hai phương thức:* 3](#_Toc110403280)

[*3.interface* ***Publisher*** *<R> có một phương thức:* 3](#_Toc110403281)

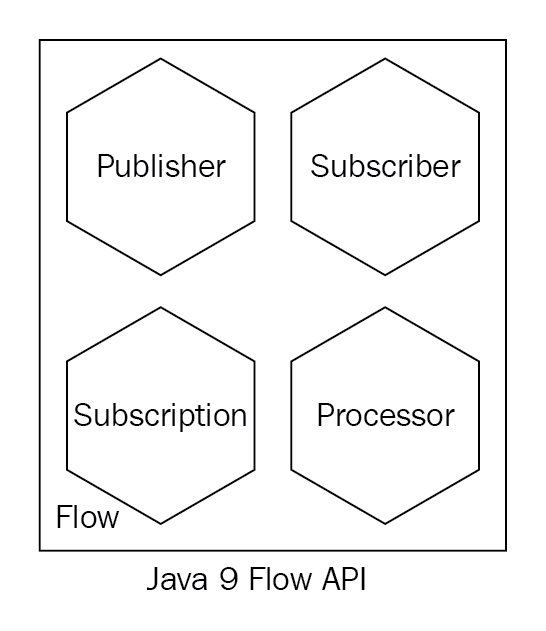
[*4.interface* ***Processor****<T,R> extends Subscriber<T>, Publisher<R>* 4](#_Toc110403282)

[*5.* 4](#_Toc110403283)

[*Flow* 4](#_Toc110403284)

[**III - So sánh Observer/Observable và Flow API** 5](#_Toc110403285)

**FLOW API**

****

# **I – Sơ lược về Flow API**

**Flow API** hỗ trợ chính thức cho đặc tả **reactive streams** kể từ java 9. Nó là sự kết hợp giữa hai mẫu Iterator và Observer patterns. **Reactive Streams** là một khái niệm định nghĩa cơ chế để xử lý stream một cách bất đồng bộ (asynchronous) với non-blocking back pressure. Back pressure ở đây chúng ta có thể hiểu nôm na có quá nhiều công việc phải xử lý cùng một lúc nên dẫn tới quá tải. Non-blocking back pressure có nghĩa là để tránh cái việc mà cùng một lúc quá nhiều công việc ập tới, trong 1 thời điểm chúng ta chỉ xử lý một số công việc nào đó thôi, khi nào xử lý xong những công việc đó thì mới nhận tiếp những công việc mới…

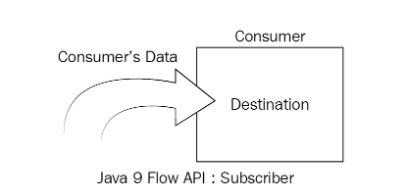
# **II – Các thành phần của Flow API** **Flow API** chứa 5 thành phần cơ bản :

1. **Subscriber<T>:** người nhận hàng từ **Publisher**
2. **Publisher:** một nhà sản xuất các mặt hàng, nguồn cung cấp các mặt hàng cho **Subscriber**.
3. **Subscription:** Kiểm soát liên kết giữa Publisher và Subscriber. ( đơn đặt hàng)
4. **Processor<T, R>** một thành phần hoạt động như một **Subscriber** và **Publisher**.
5. **Flow**

T: loại ( kiểu dữ liệu) mặt hàng đã đăng ký để nhận

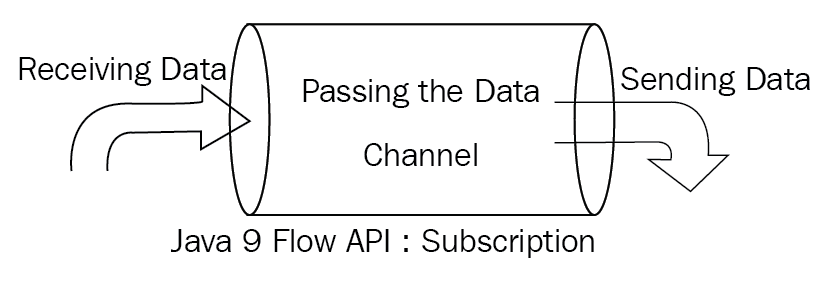
R: loại ( kiểu dữ liệu) mặt hàng được phân phối

## *1. Interface* ***Subscriber****<T> có bốn phương thức:*



* Phương thức onSubscribe(Subscription subscription ): Điều này tạo ra một đăng ký mới. Nó được gọi trước khi gọi bất kỳ phương thức **Subcriber** nào khác cho **Subcription** nhất định.
* Phương thức onComplete(): được gọi khi không có yêu cầu gọi bất kỳ phương thức **Subcriber** nào trên **Subcription** đó mà chưa bị kết thúc bởi lỗi. Sau đó không có phương thức nào của **Subcriber** được gọi bởi **Subcription** đó.
* Phương thức onError(Throwable throwable): được gọi ra khi một **Publisher** hoặc **Subscription** gặp phải lỗi không thể khôi phục được, sau đó không có phương thức **Subcriber** nào khác được **Subcription** gọi ra.
* Phương thức onNext(T item):Khi một **Subcription** được tạo, nó được gọi để nhận dữ liệu, item hoặc phần tử từ **Publisher.**

## *2.Interface* ***Subscription*** *có hai phương thức:*

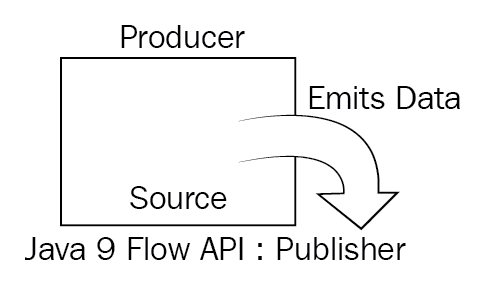


* Phương thức request(long n): Phương thức này được sử dụng để thêm n số item nhất định vào **Subcription** đang hoạt động hiện tại giữa **Publisher** và (các)**Subcriber**

Khi n <=0, Subscriber nhận được tin nhắn phương thức onError do tham số ( số lượng hàng yêu cầu) không thể bằng và bé hơn 0, ngược lại Subscriber nhận được n mặt hàng

* Phương thức cancel(): dùng để ngừng, huỷ đăng ký **Subcription** hiện tại giữa **Publisher** và các **Subcriber** vì thế không có sự giao tiếp giữa chúng nữa.

## *3.interface* ***Publisher*** *<R> có một phương thức:*



* Phương thức subscribe(Subscriber<? super T> subscriber): Phương thức này dùng để thêm 1 Subscriber để phân phối hàng cho nó. Nếu Subscriber này đã đăng ký 1 Publisher hoặc đăng ký bị lỗi ( do nhiều nguyên nhân), nó gọi (message) tới Subscriber phương thức onError. Nếu thêm thành công thì gửi cho (message) Subscriber phương thức onSubscribe.

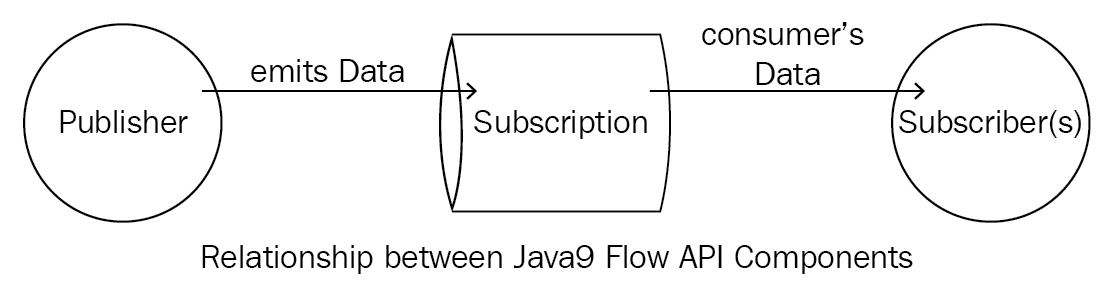
? super T: kiểu cha trừu tượng hoặc interface class của T // với T thì xem ở trên

## *4.interface* ***Processor****<T,R> extends Subscriber<T>, Publisher<R>*

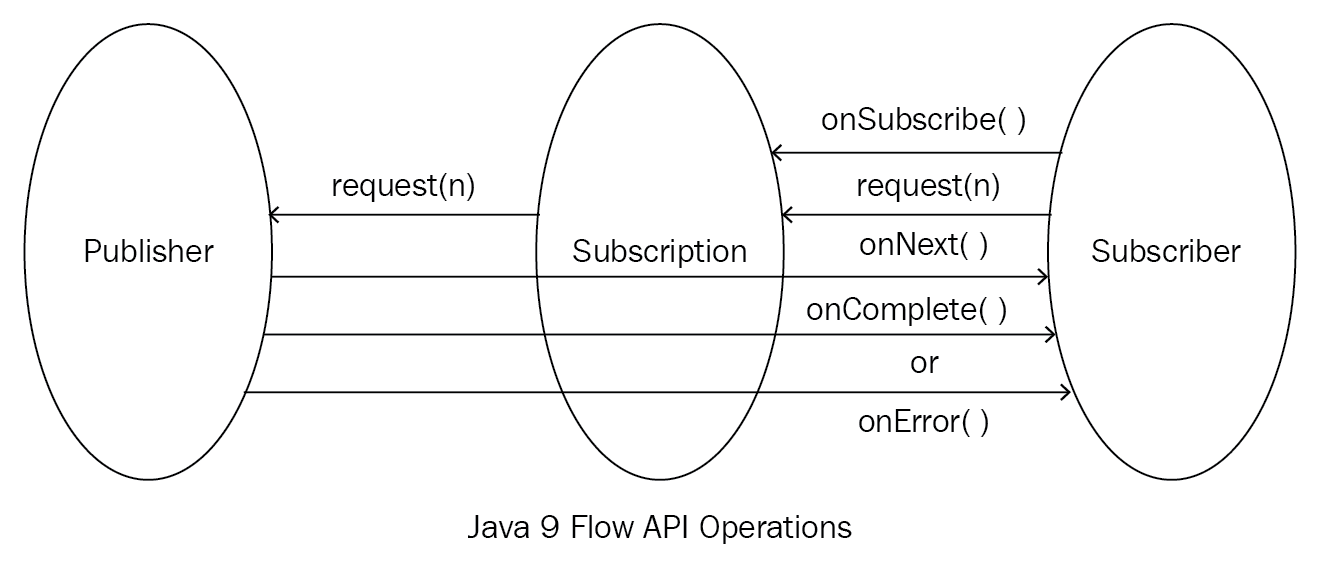
* Trong Flow API **Processor** là một thành phần đặc biệt. Nó hoạt động vừa với tư cách **Publisher** và **Subcriber** . Tức là ta có thể sử dụng nó làm nguồn dữ liệu, 1 **Publisher** hoặc đích đến nhận dữ liệu, 1 **Subcriber.**

## *5.**Flow*

* Trong các phần trước, chúng ta đã thảo luận sâu về các thành phần của Flow API. Chúng là tất cả các giao diện và được định nghĩa là các thành phần tĩnh trong một thành phần khác của Flow API. Thành phần này là Flow.
* Trong java 9, Flow chứa 4 static interface **Publisher, Subcrber, Subcription và Processor**
* Khi chúng ta kết hợp hay kết nối chúng lại với nhau trong một hệ thống chúng sẽ hoạt động như sau:



* Khi chúng ta kết nối các thành phần Flow API theo cách này, chúng ta có thể quan sát thấy rằng một luồng đang đi từ nguồn đến đích. Đó là lý do tại sao họ đã đặt tên API này là flow API.
* Chúng ta có thể trình bày cách giao tiếp hoàn chỉnh của **Publisher**/ **Subcriber** của Java 9 Flow như được hiển thị trong sơ đồ sau. Giao tiếp này có thể kết thúc thành công hoặc không thành công:



# **III - So sánh Observer/Observable và Flow API**

|  |  |  |
| --- | --- | --- |
|  | **Giống nhau** | **Khác nhau** |
| **Flow API** | - Đều là các mẫu thiết kế của Java  - Đều có các interface Publisher và Subscriber  - Đều dùng để gửi hoặc nhận tin nhắn | - Có các interface Subscription, Processor  - Implementation, liên kết lỏng lẻo với nhau  - Dễ bảo trì, cập nhật |
| **Observable/Observer** | - Extend từ các Observable, liên kết chặt chẽ với nhau.  - Kế thừa code  - Không còn được sử dụng  - Khó bảo trì hoặc sửa đổi  - Dễ truyền nhầm tham số Observable |

**Thành viên:**

1. Bùi Quang Trung
2. Bùi Thị Trọng
3. Châu Thanh Tâm
4. Nguyễn Phú