# Overview of the Java Reactive Streams API (Part 1)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

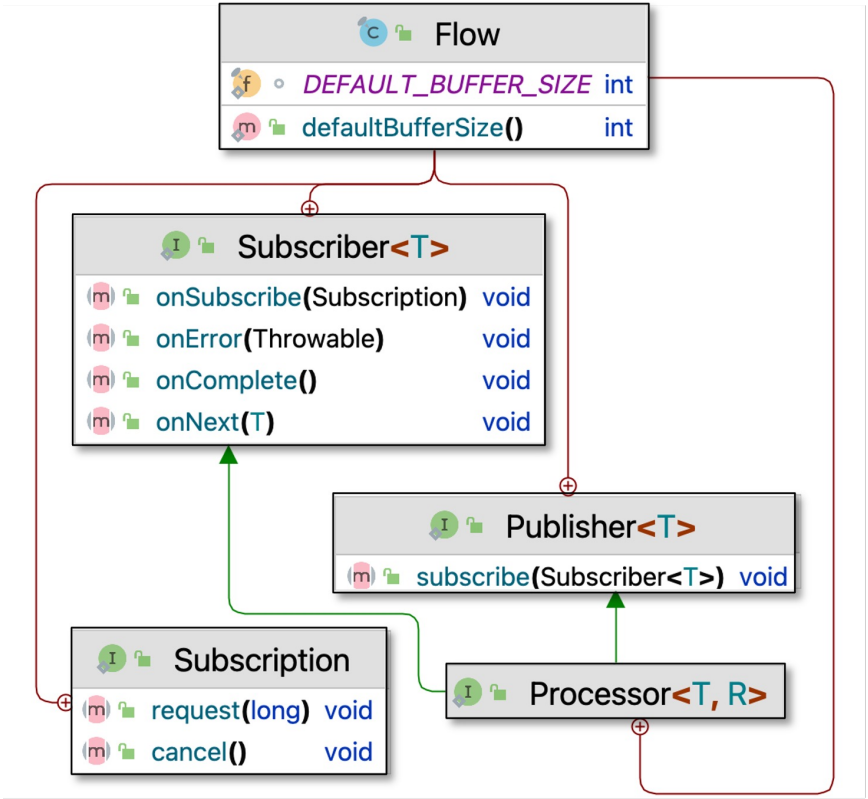
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

# Learning Objectives in this Part of the Lesson

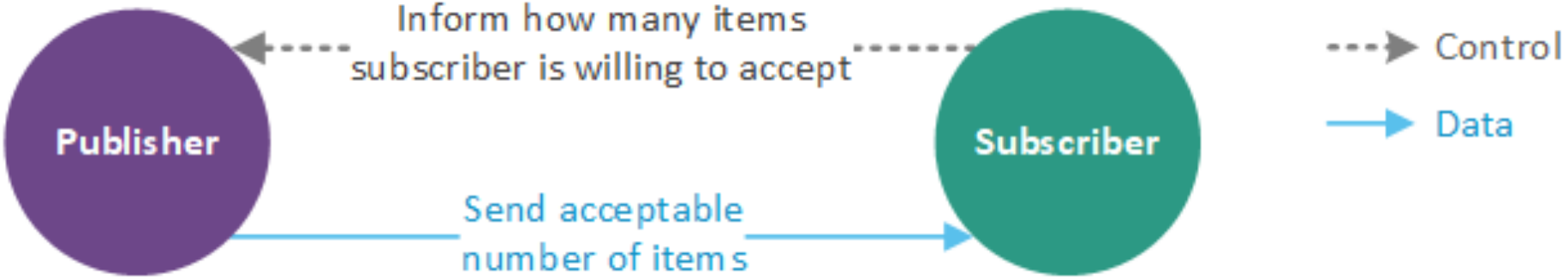
- Understand the key benefits & principles underlying the reactive programming paradigm
- Know the Java reactive streams API




See community.oracle.com/docs/DOC-1006738

# Learning Objectives in this Part of the Lesson

- Understand the key benefits & principles underlying the reactive programming paradigm
- Know the Java reactive streams API
  - Be aware of key patterns




See medium.com/@mahdichtioui/reactivex-reactive-programming-principles-dbb1bafa8384

# Java Reactive Streams API & Key Patterns

# Java Reactive Streams API & Key Patterns

- Java 9+ supports reactive programming via Reactive Streams & the Flow API

## Class Flow

java.lang.Object
    java.util.concurrent.Flow

---

```
public final class Flow
extends Object
```

Interrelated interfaces and static methods for establishing flow-controlled components in which `Publishers` produce items consumed by one or more `Subscribers`, each managed by a `Subscription`.

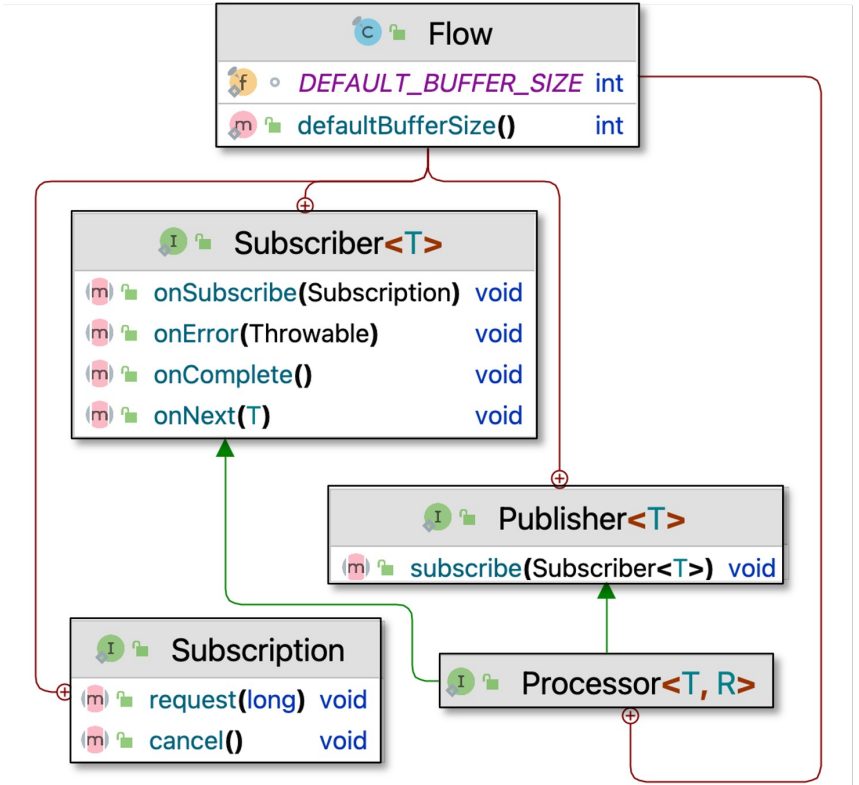
These interfaces correspond to the reactive-streams specification. They apply in both concurrent and distributed asynchronous settings: All (seven) methods are defined in `void` "one-way" message style. Communication relies on a simple form of flow control (method `Flow.Subscription.request(long)`) that can be used to avoid resource management problems that may otherwise occur in "push" based systems.

**Examples.** A `Flow.Publisher` usually defines its own `Flow.Subscription` implementation; constructing one in method `subscribe` and issuing it to the calling `Flow.Subscriber`. It publishes items to the subscriber asynchronously, normally using an `Executor`. For example, here is a very simple publisher that only issues (when requested) a single `TRUE` item to a single subscriber. Because the subscriber receives only a single item, this class does not use buffering and ordering control required in most implementations (for example `SubmissionPublisher`).

See docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.html

# Java Reactive Streams API & Key Patterns

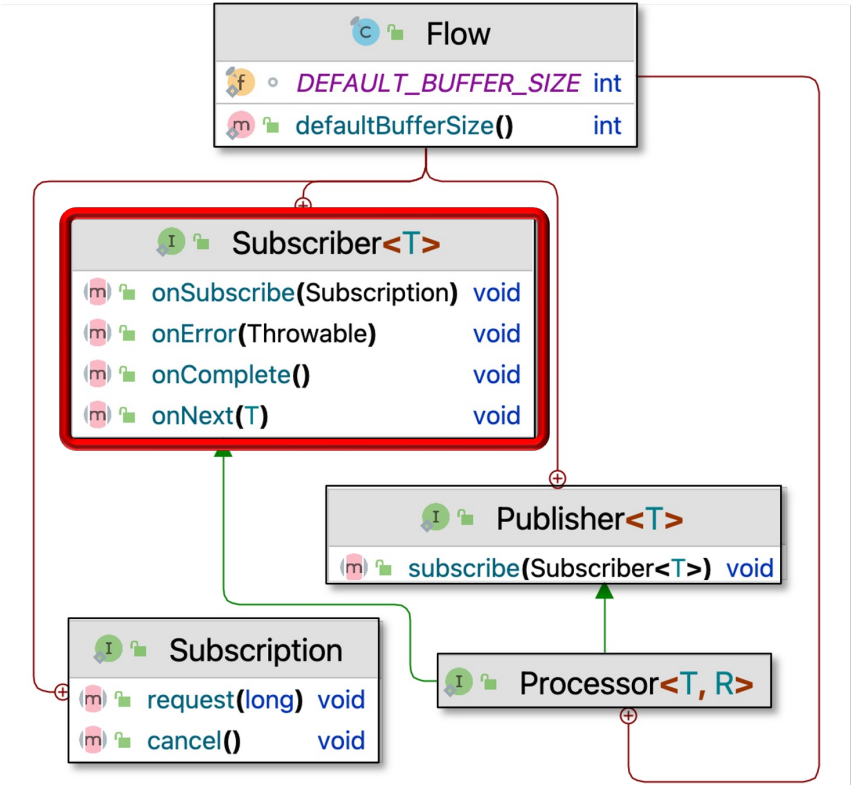
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations




See www.reactive-streams.org

# Java Reactive Streams API & Key Patterns

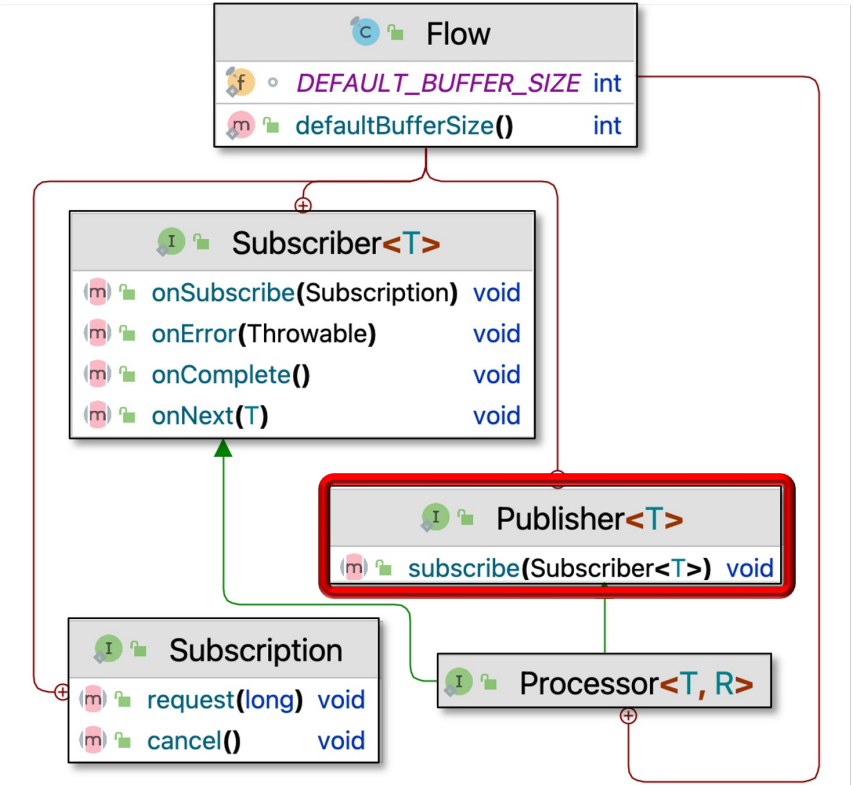
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Subscriber**
    - Defines methods that a data receiver must implement to get data from a Publisher
      - These methods include handling subscription, error events, completion, & data arrival




See docs.oracle.com/javase/java/util/concurrent/Flow.Subscriber.html

# Java Reactive Streams API & Key Patterns

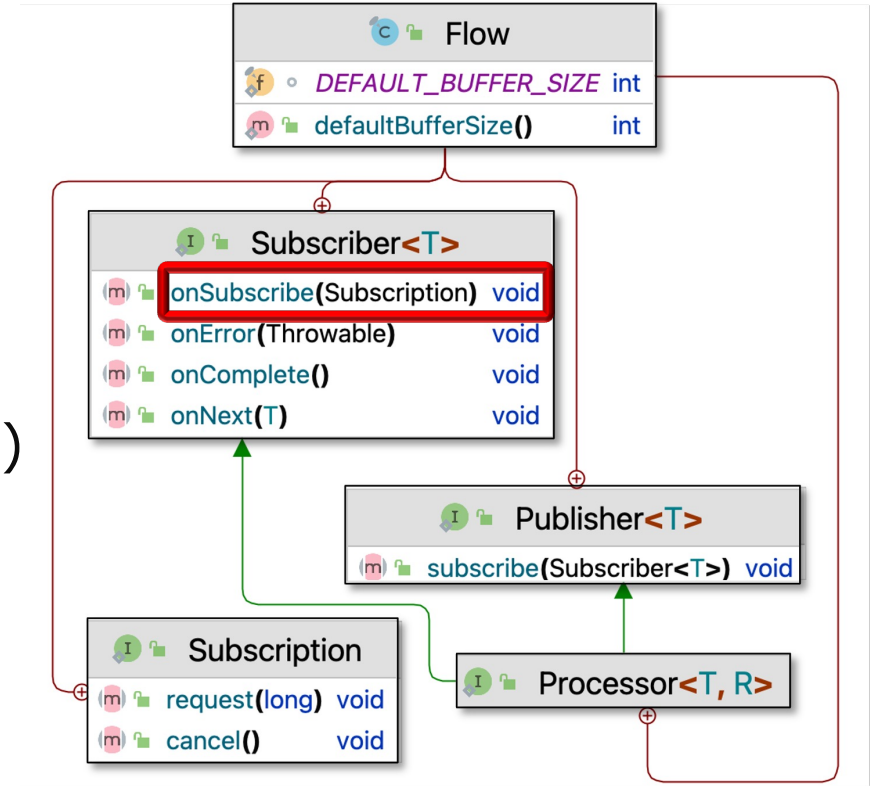
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Publisher**
    - An interface representing a data source that can be subscribed to by Subscribers




See docs.oracle.com/javase/java/util/concurrent/Flow.Publisher.html

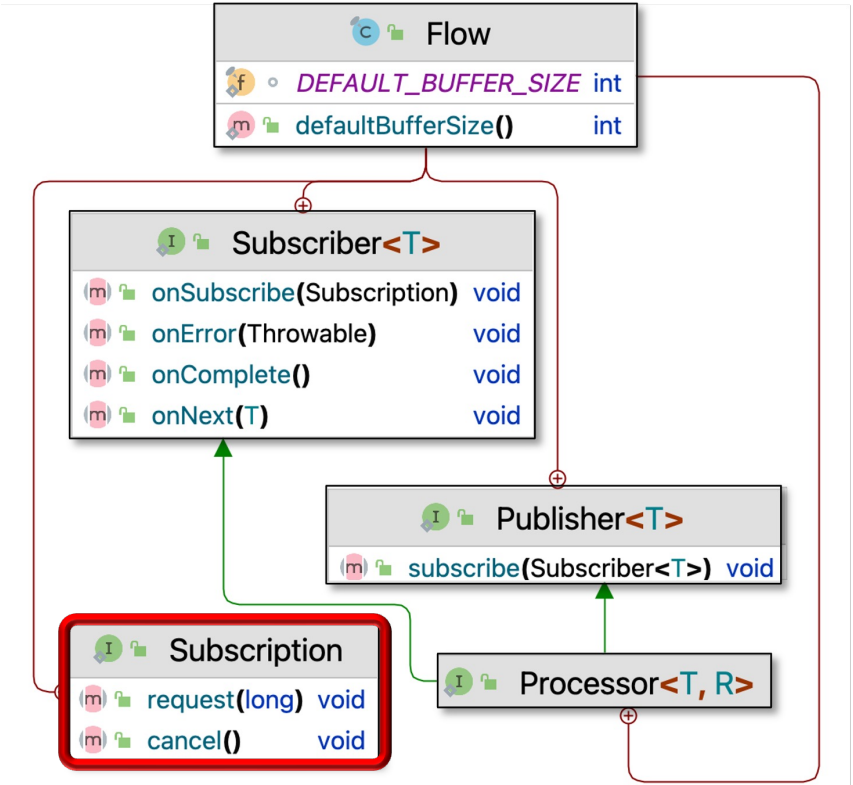
# Java Reactive Streams API & Key Patterns

- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Publisher**
    - An interface representing a data source that can be subscribed to by Subscribers
      - After a Subscriber subscribes, the Publisher calls onSubscribe() to provide a Subscription



# Java Reactive Streams API & Key Patterns

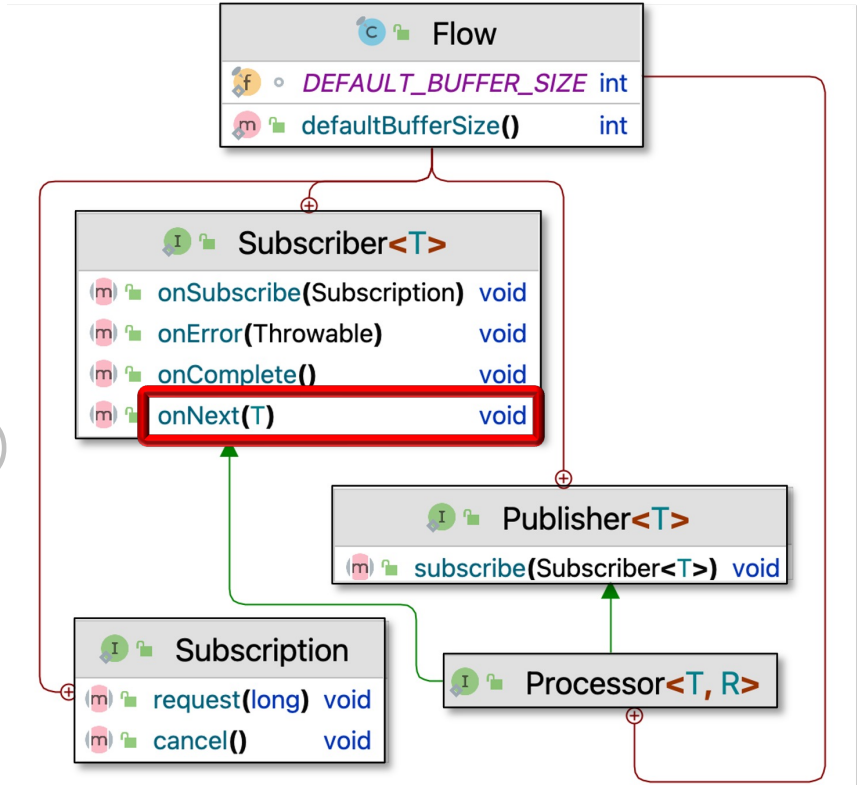
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Subscription**
    - A link between a Publisher & Subscriber
      - It allows the Subscriber to control the flow of data from the Publisher by requesting more data or cancelling the subscription




See docs.oracle.com/javase/java/util/concurrent/Flow.Subscription.html

# Java Reactive Streams API & Key Patterns

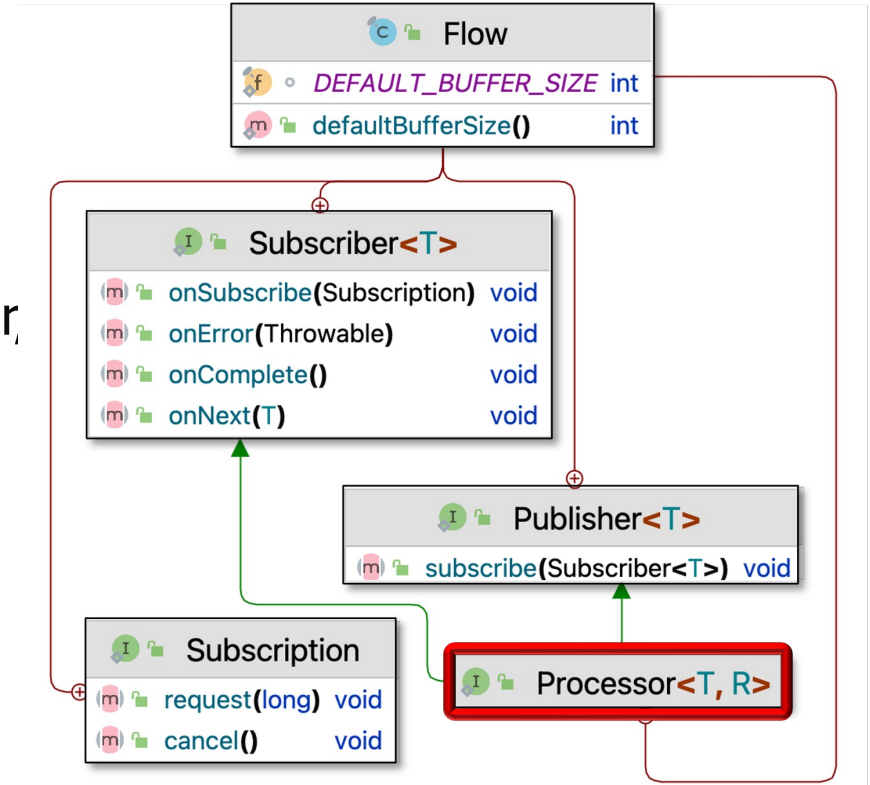
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Publisher**
    - An interface representing a data source that can be subscribed to by Subscribers
      - After a Subscriber subscribes, the Publisher calls onSubscribe() to provide a Subscription
    - The Publisher provides data by calling the Subscriber onNext(T) hook method




See docs.oracle.com/javase/java/util/concurrent/Flow.Publisher.html

# Java Reactive Streams API & Key Patterns

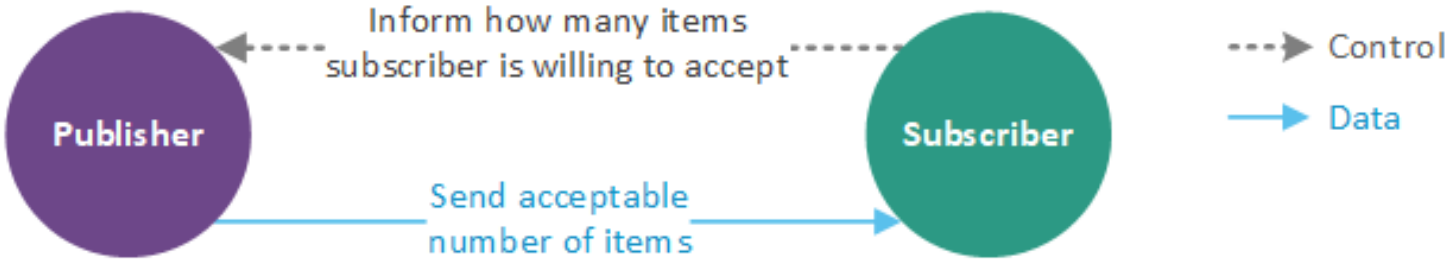
- The Java Flow API defines interfaces designed to ensure interoperability of reactive streams implementations
  - **Processor**
    - Acts both as a Subscriber & Publisher
      - It receives data from a Publisher, processes it, & publishes the results to its Subscribers




See docs.oracle.com/javase/java/util/concurrent/Flow.Processor.html

# Java Reactive Streams API & Key Patterns

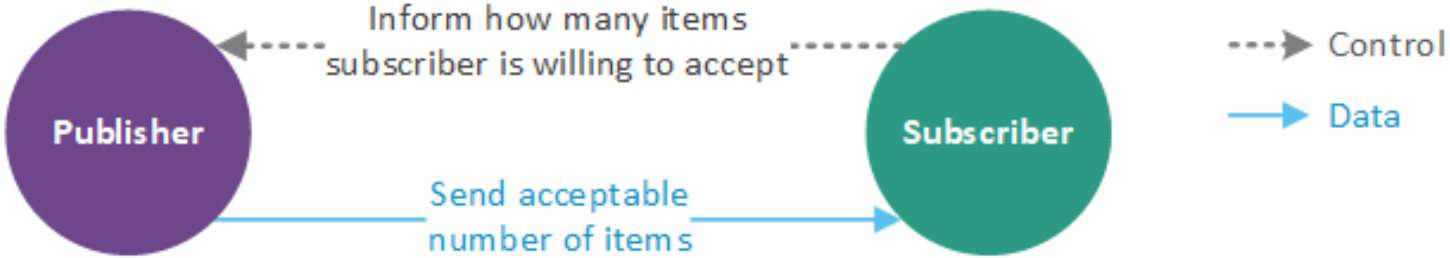
- The Flow API supports stream-oriented publish/subscribe patterns



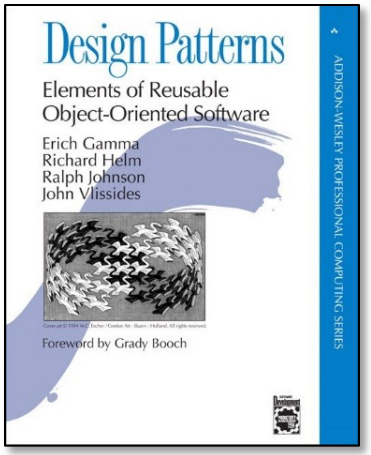

See javasampleapproach.com/java/java-9/java-9-flow-api-example-publisher-and-subscriber

# Java Reactive Streams API & Key Patterns

- The Flow API supports stream-oriented publish/subscribe patterns



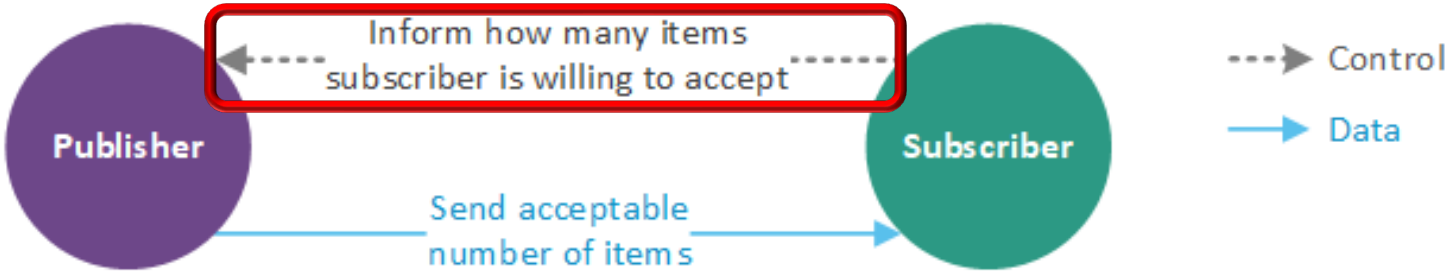

  - Combines two patterns



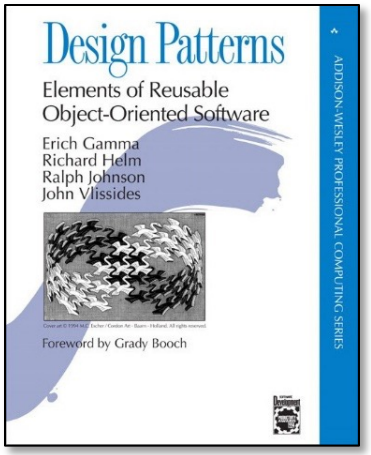

See www.journaldev.com/20723/java-9-reactive-streams

# Java Reactive Streams API & Key Patterns

- The Flow API supports stream-oriented publish/subscribe patterns




  - Combines two patterns
    - *Iterator*, which applies a "pull model" where app subscriber(s) pull items from a publisher source

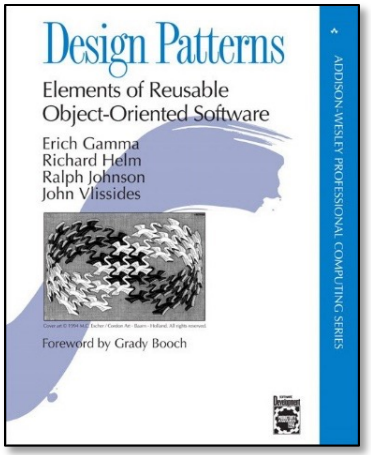



See en.wikipedia.org/wiki/Iterator_pattern

# Java Reactive Streams API & Key Patterns

- The Flow API supports stream-oriented publish/subscribe patterns

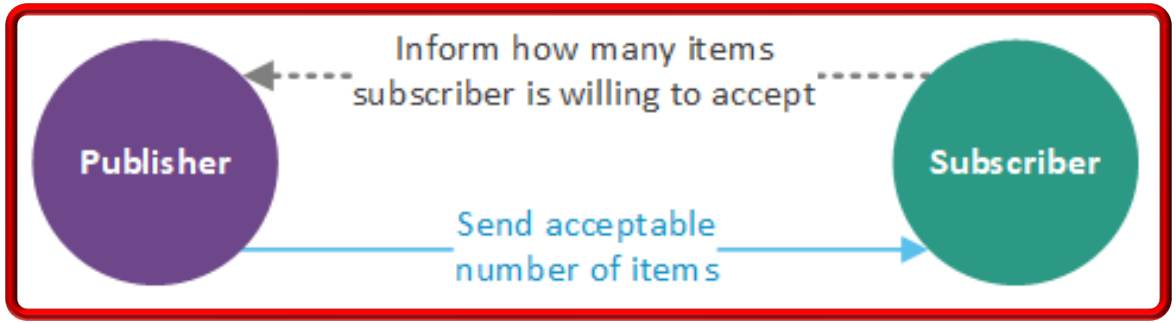

  - Combines two patterns
    - *Iterator*, which applies a "pull model" where app subscriber(s) pull items from a publisher source
    - *Observer*, which applies a "push model" that reacts when a publisher source pushes an item to subscriber sink(s)



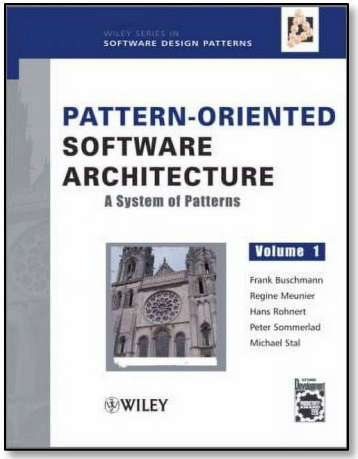

See en.wikipedia.org/wiki/Observer_pattern

# Java Reactive Streams API & Key Patterns

- The Flow API supports stream-oriented publish/subscribe patterns




  - Combines two patterns
  - Yields the Publisher/Subscriber pattern
    - Enable publisher(s) to announce events to interested subscriber(s) asynchronously, without tightly coupling publishers with subscribers




See en.wikipedia.org/wiki/Publish-subscribe_pattern

# End of Overview of the Java Reactive Streams API (Part 1)