Contrasting Java Streams with Other Technologies

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

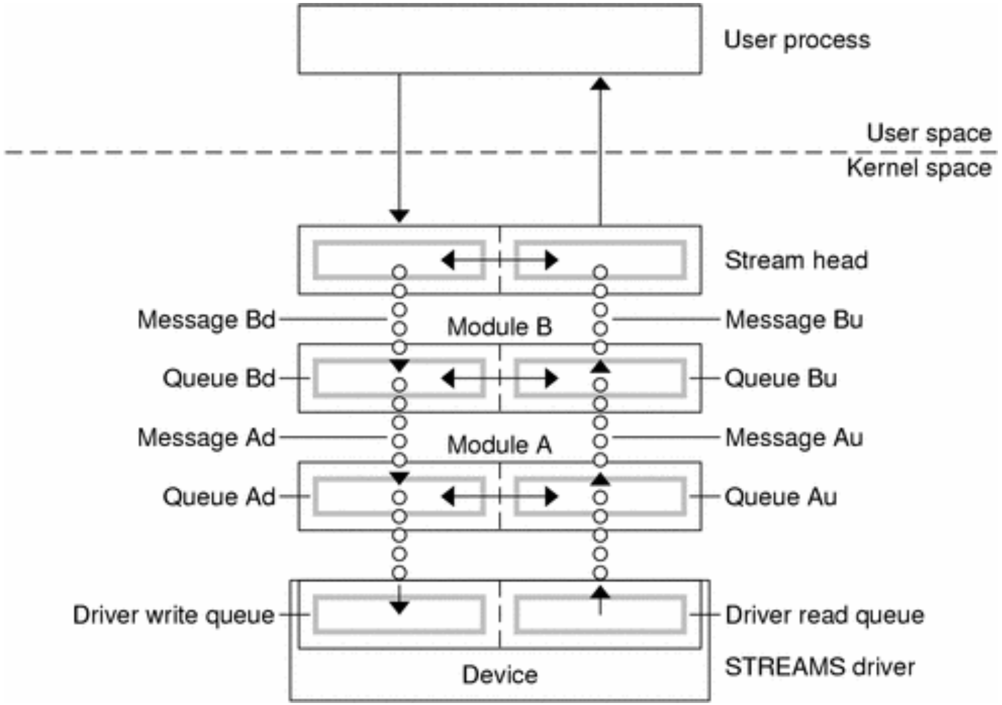
Professor of Computer Science

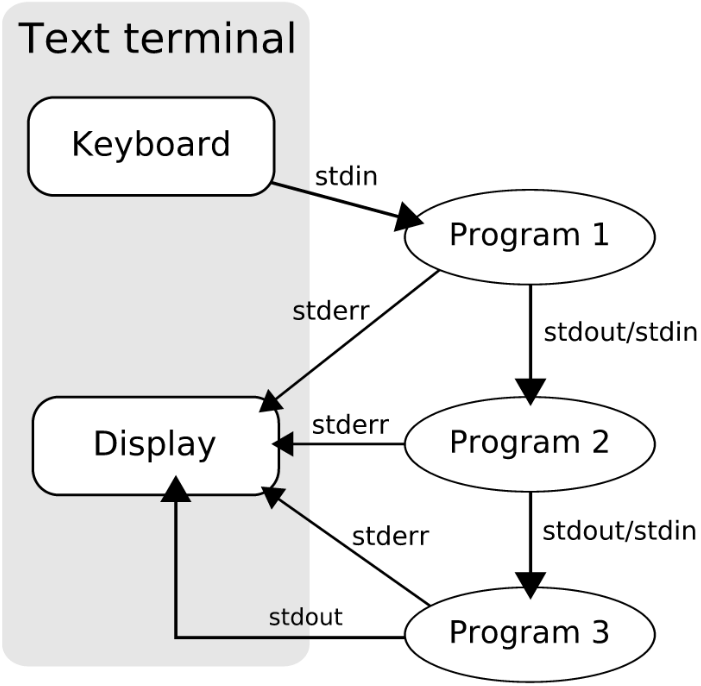
Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how Java streams compare with other technologies
  - e.g., UNIX pipelines, System V STREAMS, & Java Reactive Streams

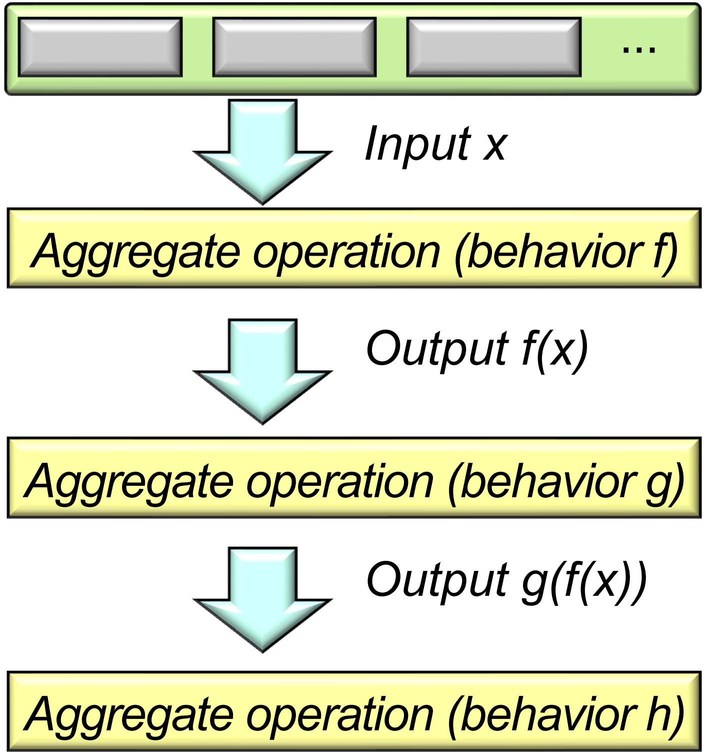




Contrasting Java Streams with Other Technologies

Review of Java Streams

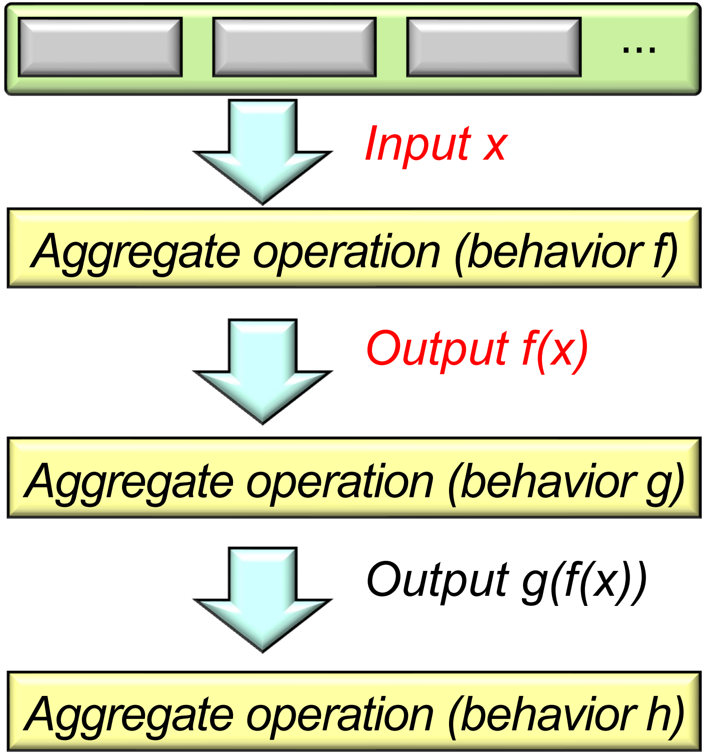
- A Java stream is a sequence of data items that are conceptually produced one at a time


…
Input x
Aggregate operation (behavior f)
Output f(x)
Aggregate operation (behavior g)
Output g(f(x))
Aggregate operation (behavior h)

See docs.oracle.com/javase/tutorial/collections/streams

Review of Java Streams

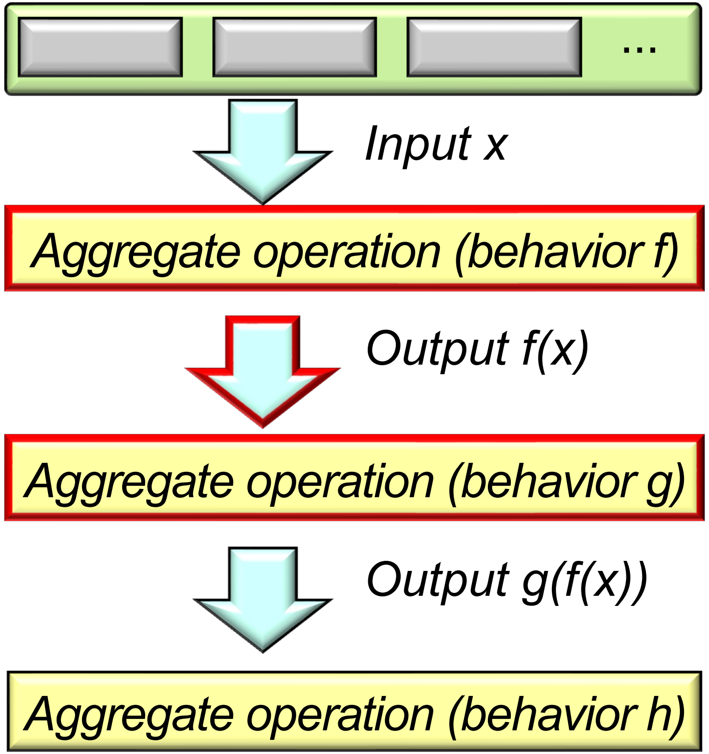
- A Java stream is a sequence of data items that are conceptually produced one at a time
  - An operation can read items from an input stream one-by-one & also write items to an output stream



…
Input x
Aggregate operation (behavior f)
Output f(x)
Aggregate operation (behavior g)
Output g(f(x))
Aggregate operation (behavior h)

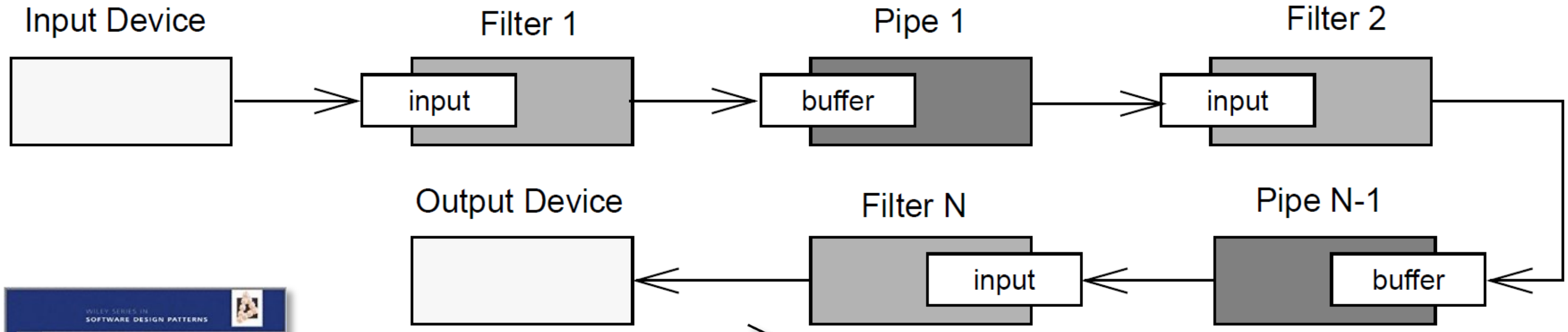
Review of Java Streams

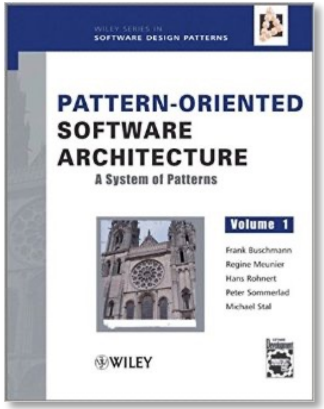
- A Java stream is a sequence of data items that are conceptually produced one at a time
  - An operation can read items from an input stream one-by-one & also write items to an output stream
  - The output stream of one operation can be the input stream of another


…
Input x
Aggregate operation (behavior f)
Output f(x)
Aggregate operation (behavior g)
Output g(f(x))
Aggregate operation (behavior h)

Comparing Java Streams with Other Technologies

- A Java stream is an implementation of the POSA1 *Pipes & Filters* pattern




WILEY SERIES IN SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A System of Patterns

Volume 1

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal

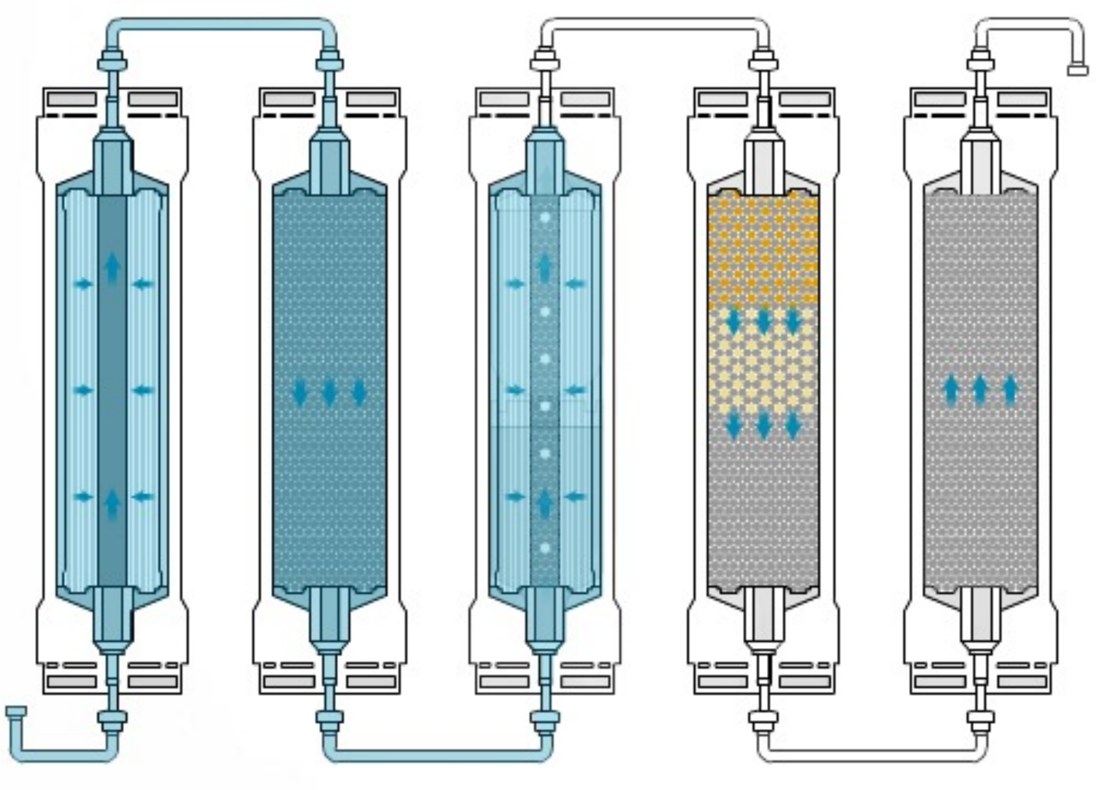
WILEY

Divide an app's tasks into multiple self-contained data processing steps & connect these steps via intermediate data buffers to form a data processing pipeline

See hillside.net/plop/2011/papers/B-10-Hanmer.pdf

Comparing Java Streams with Other Technologies

- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems



See en.wikipedia.org/wiki/Water_filter#Point-of-use_filters

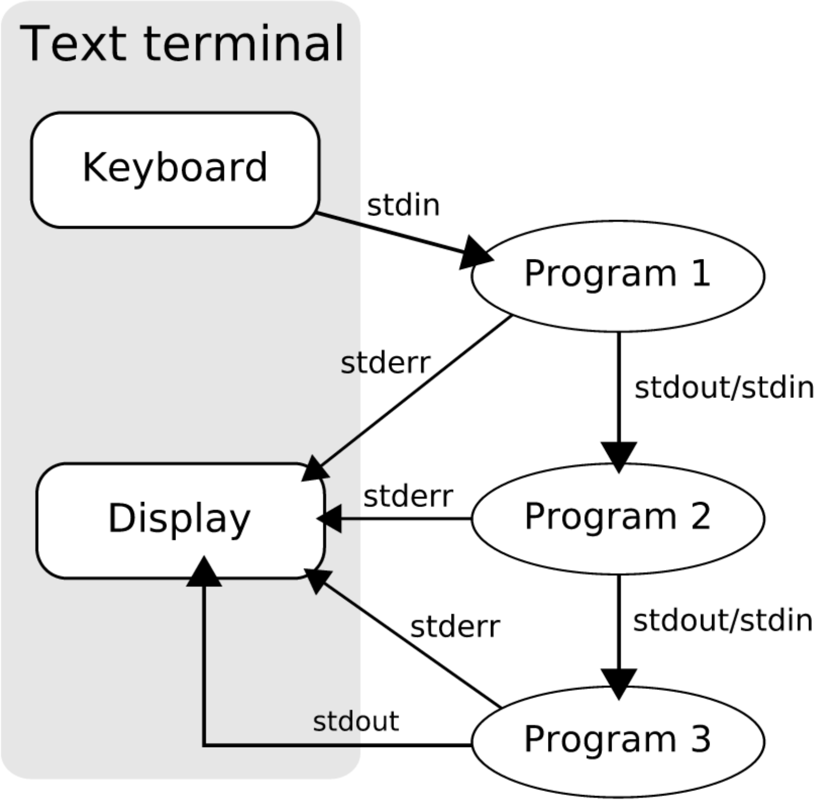
Comparing Java Streams with Other Technologies

- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems
  - A pipeline in UNIX command-line shells

```
find /usr/bin | #produce list of files
sed 's:.*/::' | #strip directory part
grep -i '^z'  | #select ‘z*’ names
sort          | #sort items
xargs -d '\n'   #print as single line
```

outputs

```
zip zipcloak zipgrep zipinfo zipnote …
```



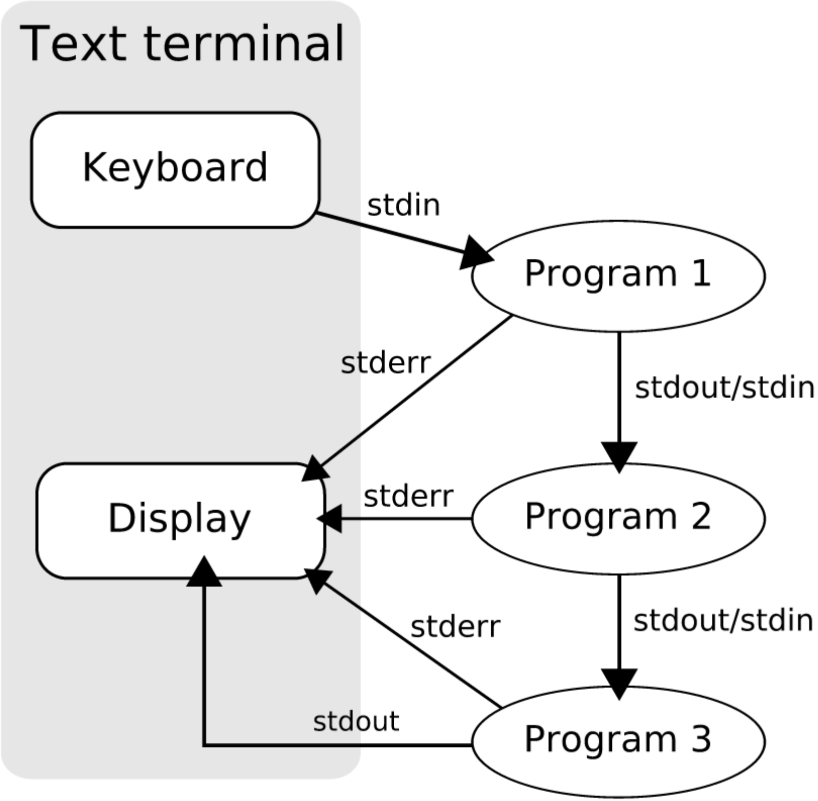
See en.wikipedia.org/wiki/Pipeline_(Unix)

Comparing Java Streams with Other Technologies

- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems
  - A pipeline in UNIX command-line shells

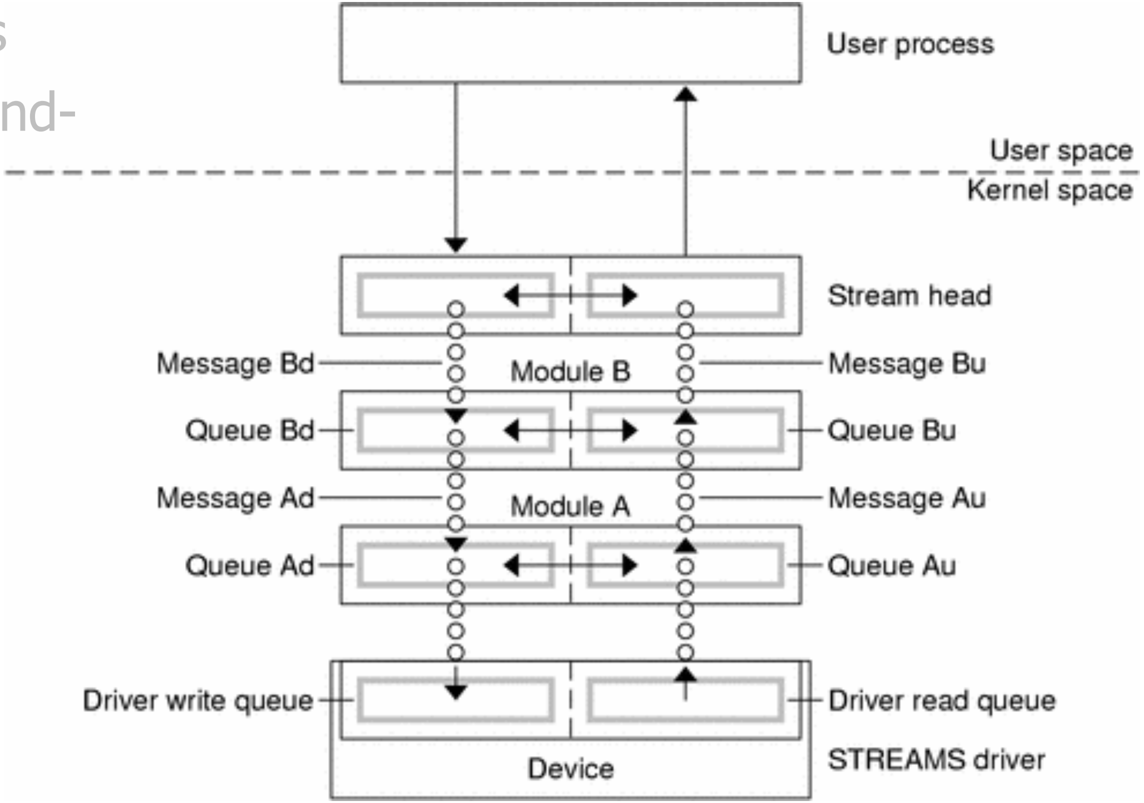
```
find /usr/bin |  #produce list of files
sed 's:.*/::' |  #strip directory part
grep -i '^z'  |  #select 'z*' names
sort          |  #sort items
xargs -d '\n'    #print as single line
```

Unlike some Java streams, a filter in a UNIX pipeline processes all contents it gets



Comparing Java Streams with Other Technologies

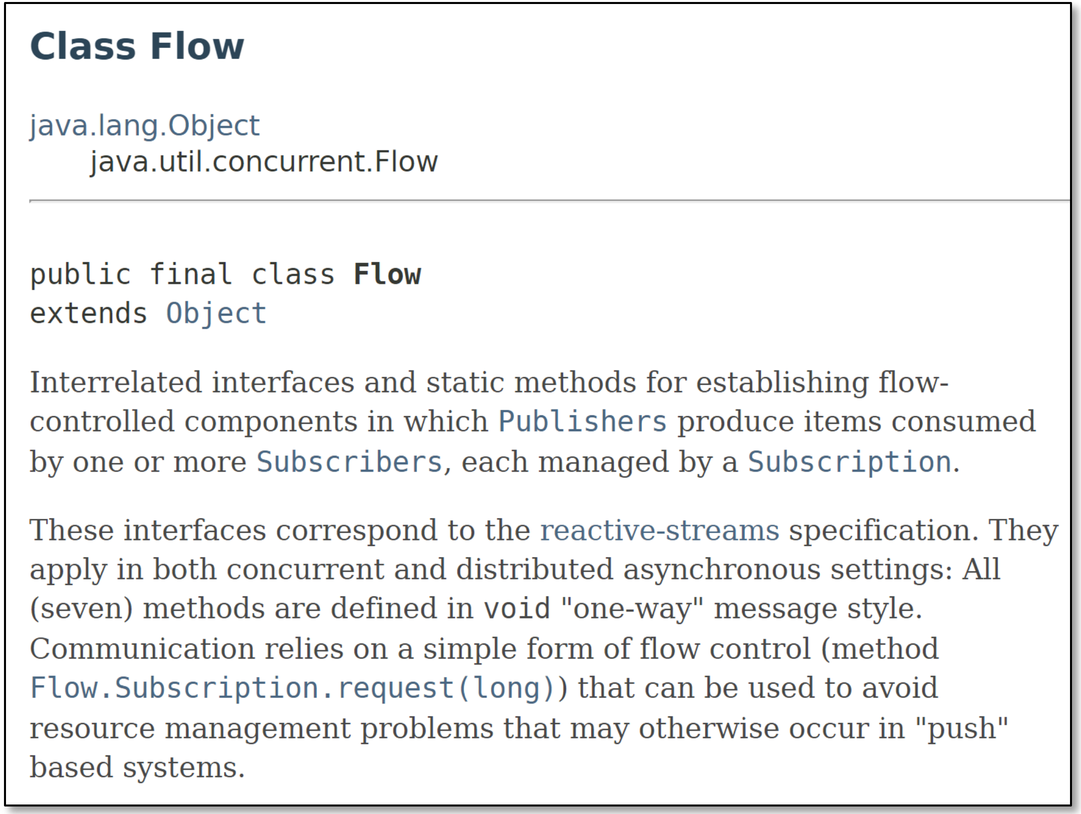
- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems
  - A pipeline in UNIX command-line shells
  - System V STREAMS



See en.wikipedia.org/wiki/STREAMS

Comparing Java Streams with Other Technologies

- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems
  - A pipeline in UNIX command-line shells
  - System V STREAMS
  - Java Reactive Streams



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`.

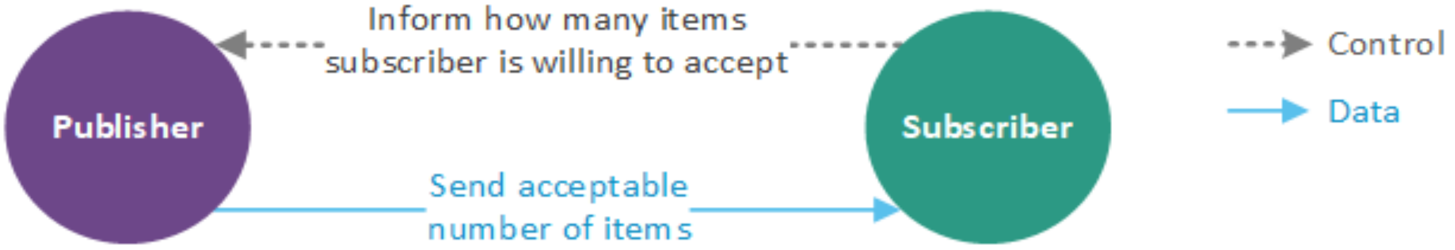
These interfaces correspond to the reactive-streams specification. They apply in both concurrent and distributed asynchronous settings: All (seven) methods are defined in `void` "one-way" message style. Communication relies on a simple form of flow control (method `Flow.Subscription.request(long)`) that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

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

Comparing Java Streams with Other Technologies

- There are other common implementations of *Pipes & Filters*, e.g.
  - Water purification systems
  - A pipeline in UNIX command-line shells
  - System V STREAMS
  - Java Reactive Streams
    - Provides an interoperable foundation for reactive programming frameworks

Project Reactor


Inform how many items subscriber is willing to accept
Publisher
Subscriber
Send acceptable number of items
Control
Data

See www.baeldung.com/java-9-reactive-streams

End of Contrasting Java Streams with Other Technologies