Contrasting Java 8 Streams with Other Technologies & Java Libraries

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

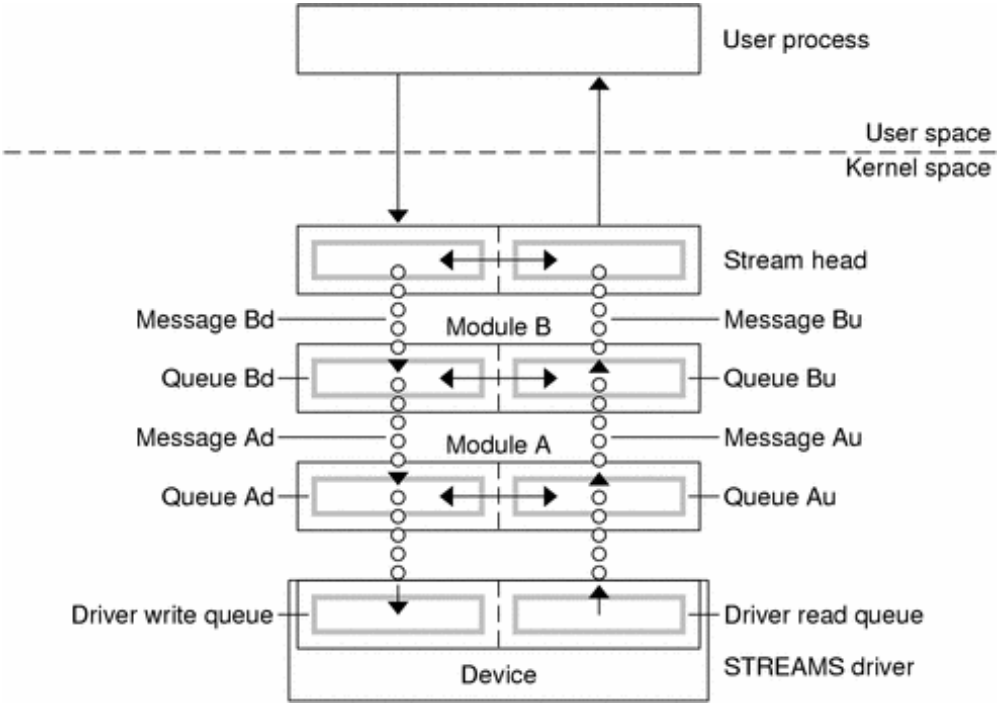
Professor of Computer Science

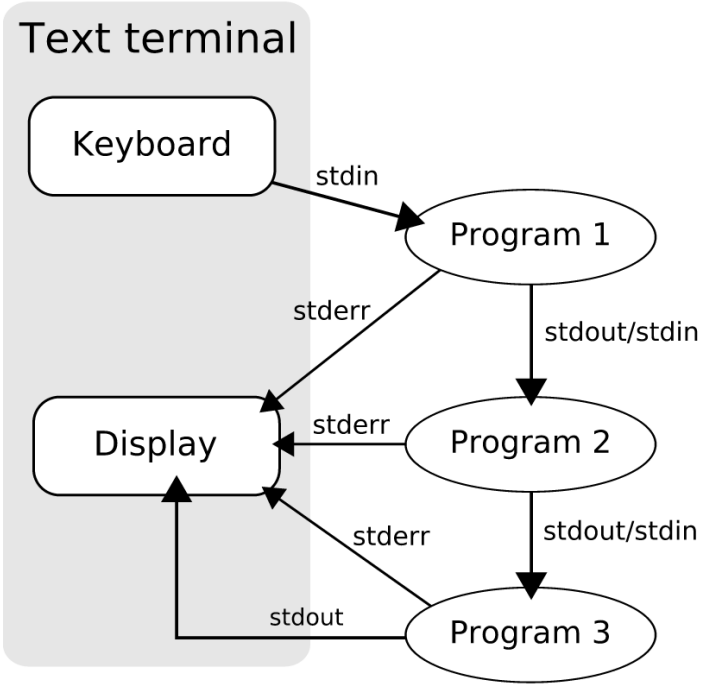
Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

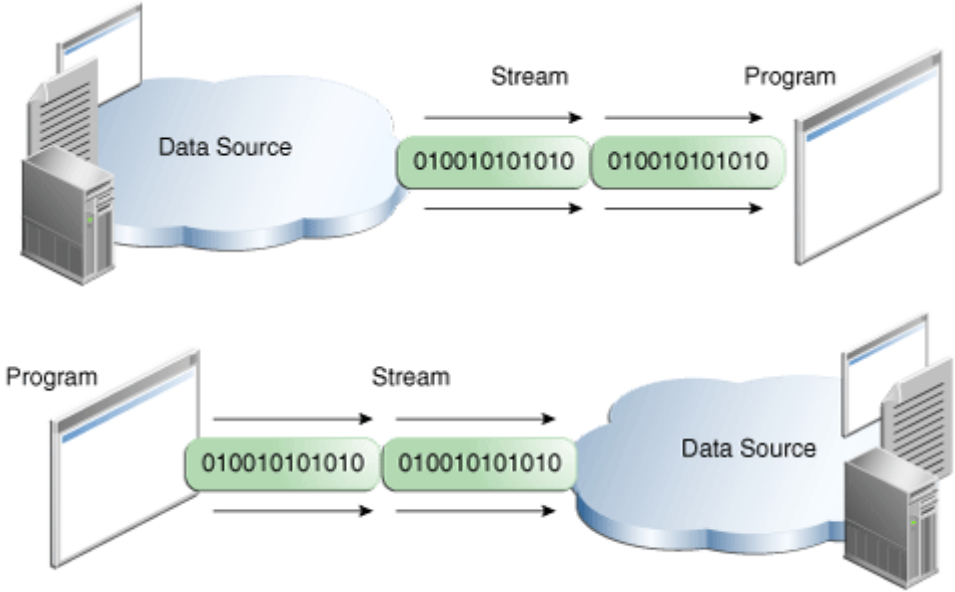
- Understand how Java 8 streams compare with other technologies

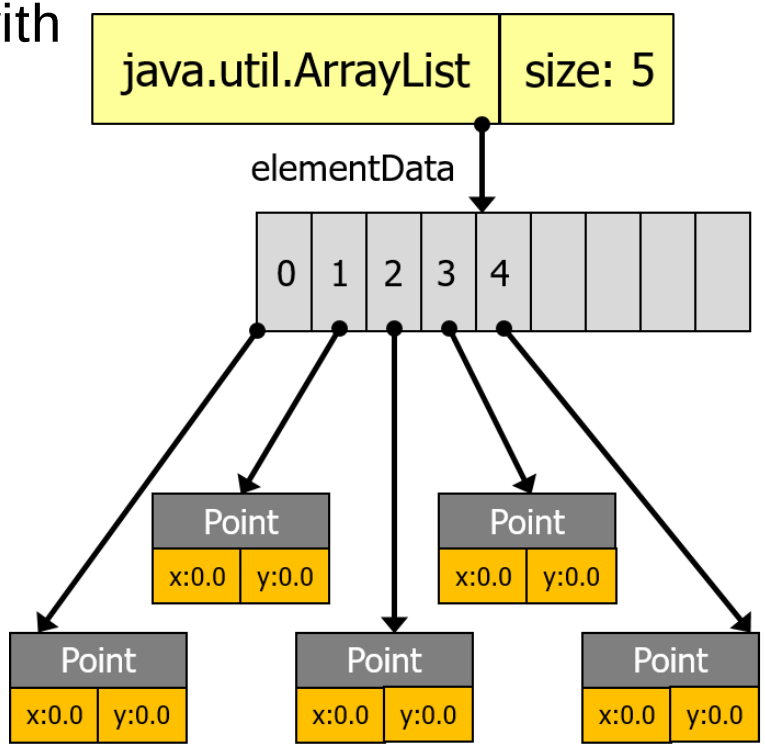




Learning Objectives in this Lesson

- Understand how Java 8 streams compare with other technologies
- Understand how Java 8 streams compare with other Java libraries

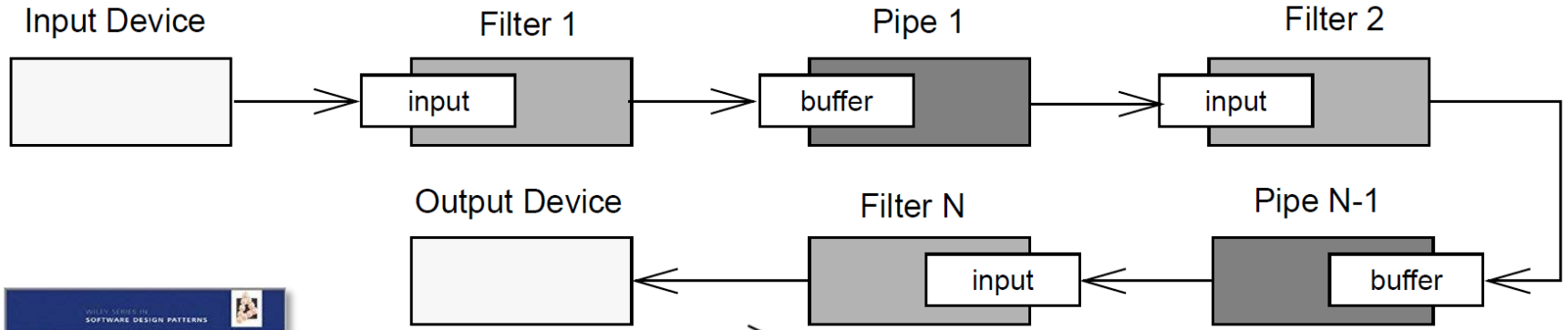


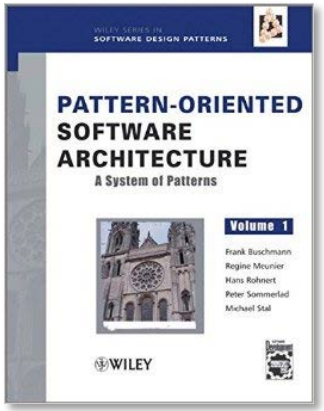


Contrasting Java 8 Streams with Other Technologies

Overview of Java 8 Streams

- A Java 8 stream is an implementation of the POSA1 *Pipes & Filters* pattern



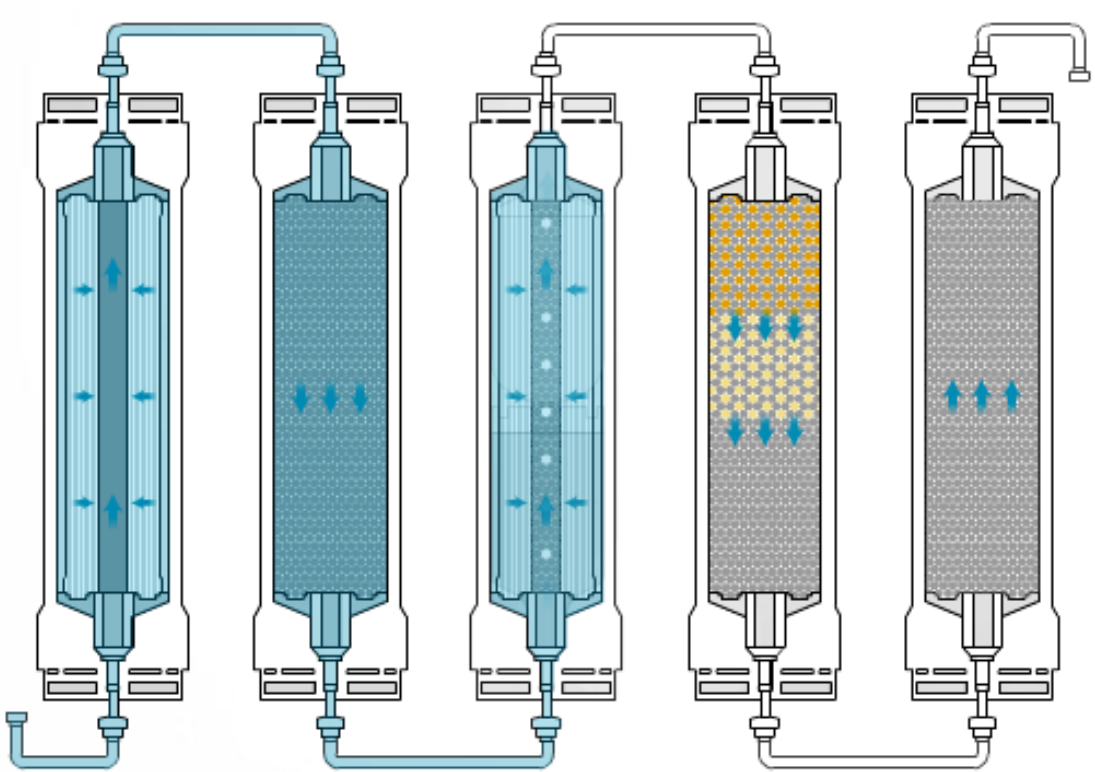


Divide an app's tasks into multiple self-contained data processing steps & connect these steps via intermediate data buffers to form a data processing pipeline

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

Overview of Java 8 Streams

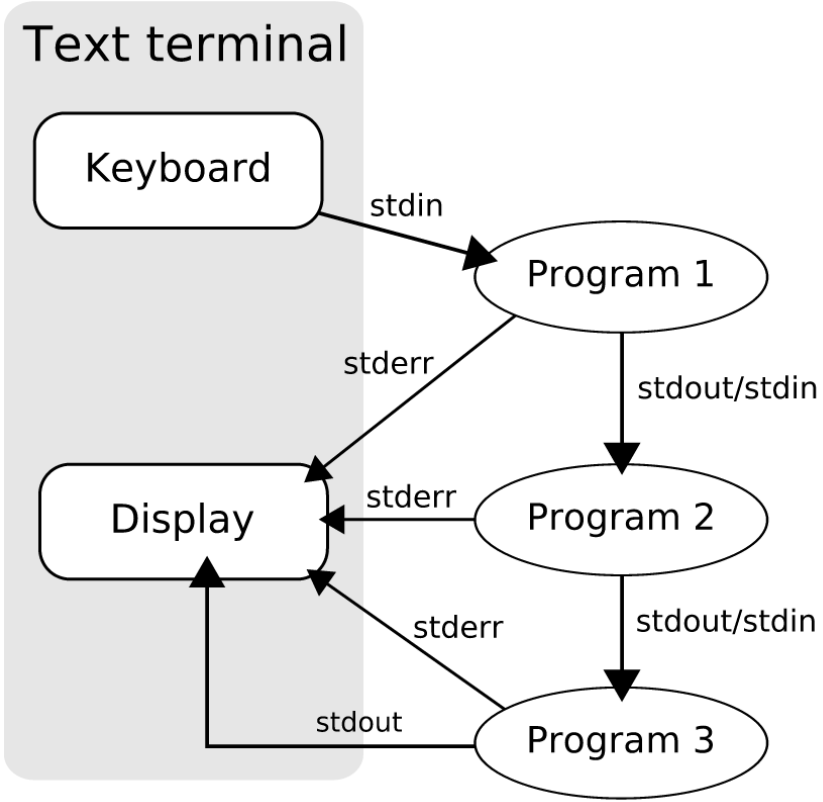
- There are other common implementations of *Pipes & Filters*



Overview of Java 8 Streams

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

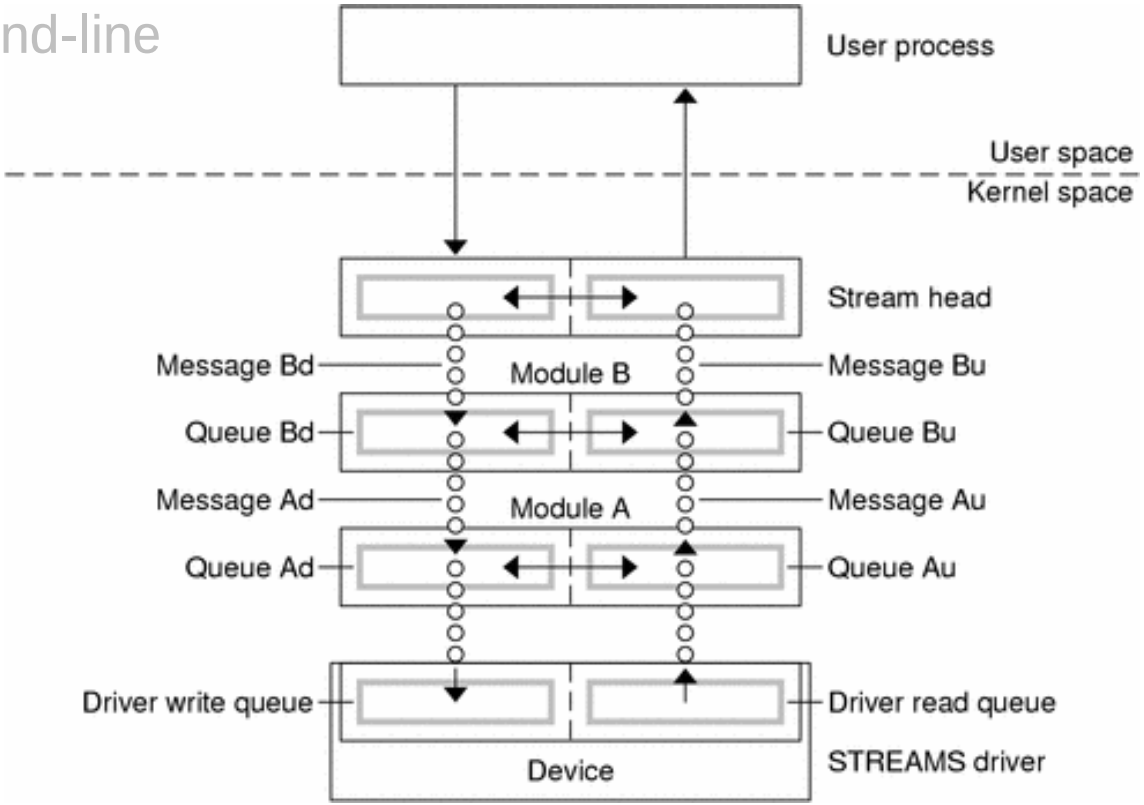
```
find /usr/bin | #produce
sed 's:.*/::' | #strip directory part
grep -i '^z'  | #select certain items
sort          | #sort items
xargs -d '\n'   #print as single line
```



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

Overview of Java 8 Streams

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

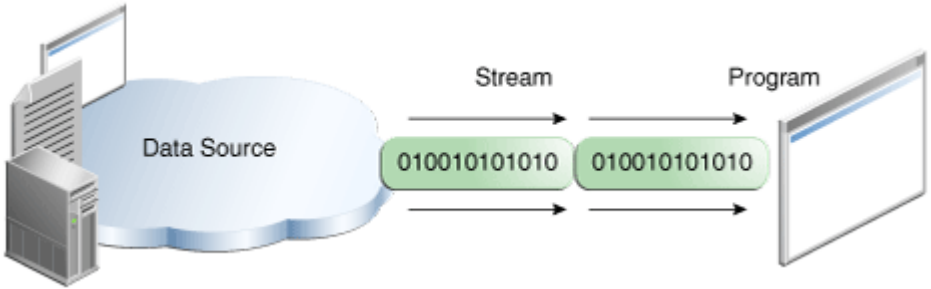


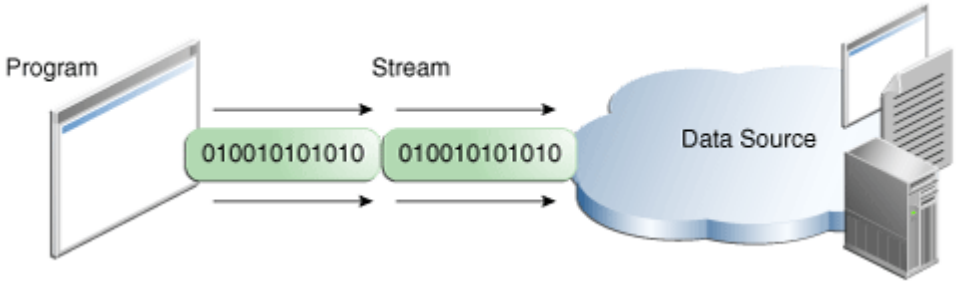
See en.wikipedia.org/wiki/STREAMS

Contrasting Java 8 Streams with Other Java Libraries

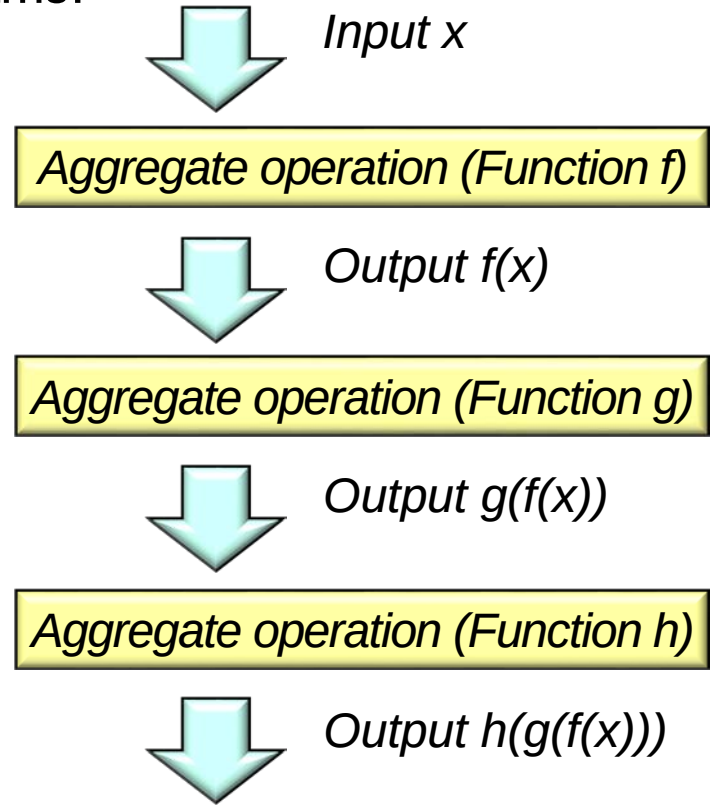
Contrasting Java I/O Streams & Java 8 Streams

- Java I/O streams are different from Java 8 streams!





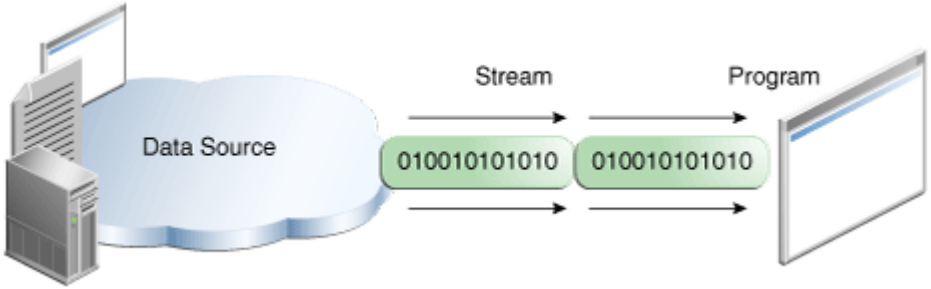
≠

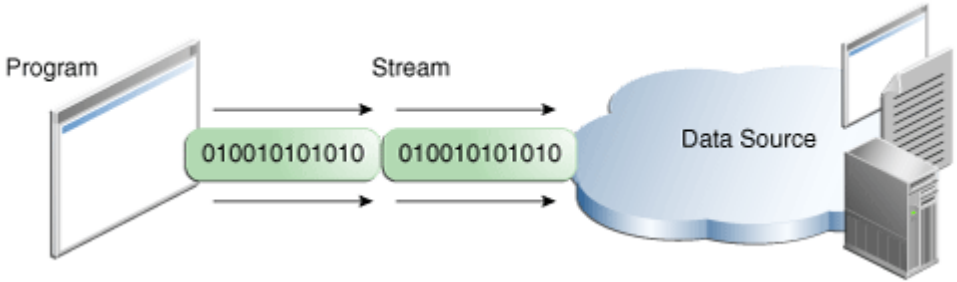


See docs.oracle.com/javase/tutorial/essential/io/streams.html

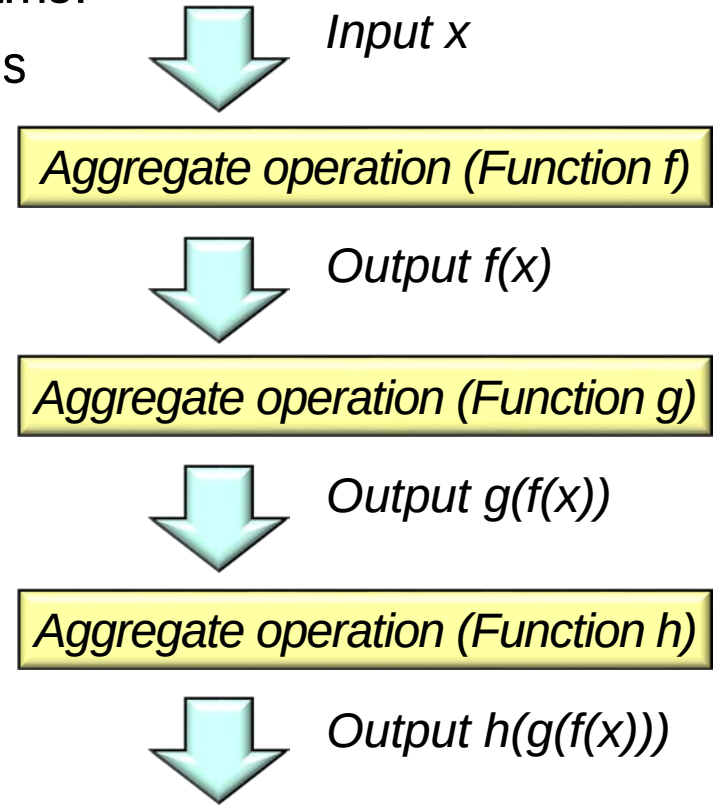
Contrasting Java I/O Streams & Java 8 Streams

- Java I/O streams are different from Java 8 streams!
  - They are often used together in Java programs



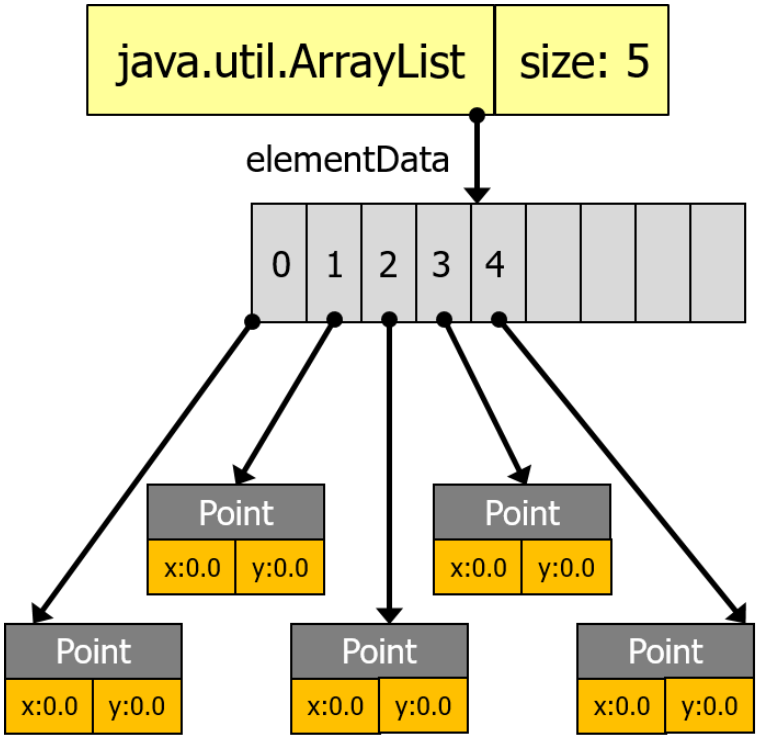


≠

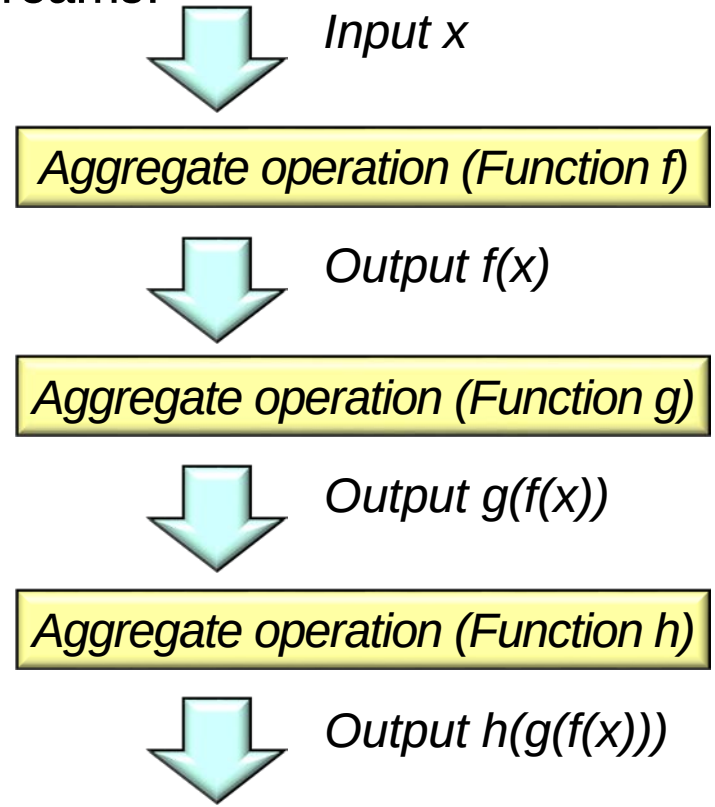


Contrasting Collections & Streams

- Java collections are also different from Java 8 streams!



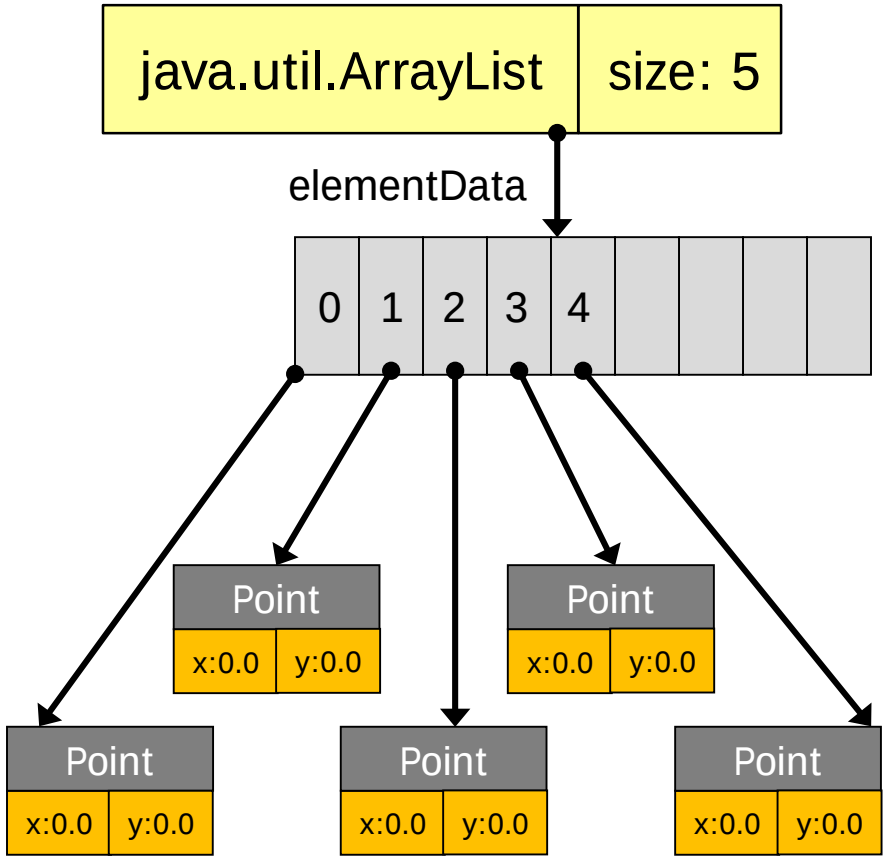
≠



See www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

Contrasting Collections & Streams

- A collection is an in-memory data structure that can store, retrieve, & manipulate groups of elements



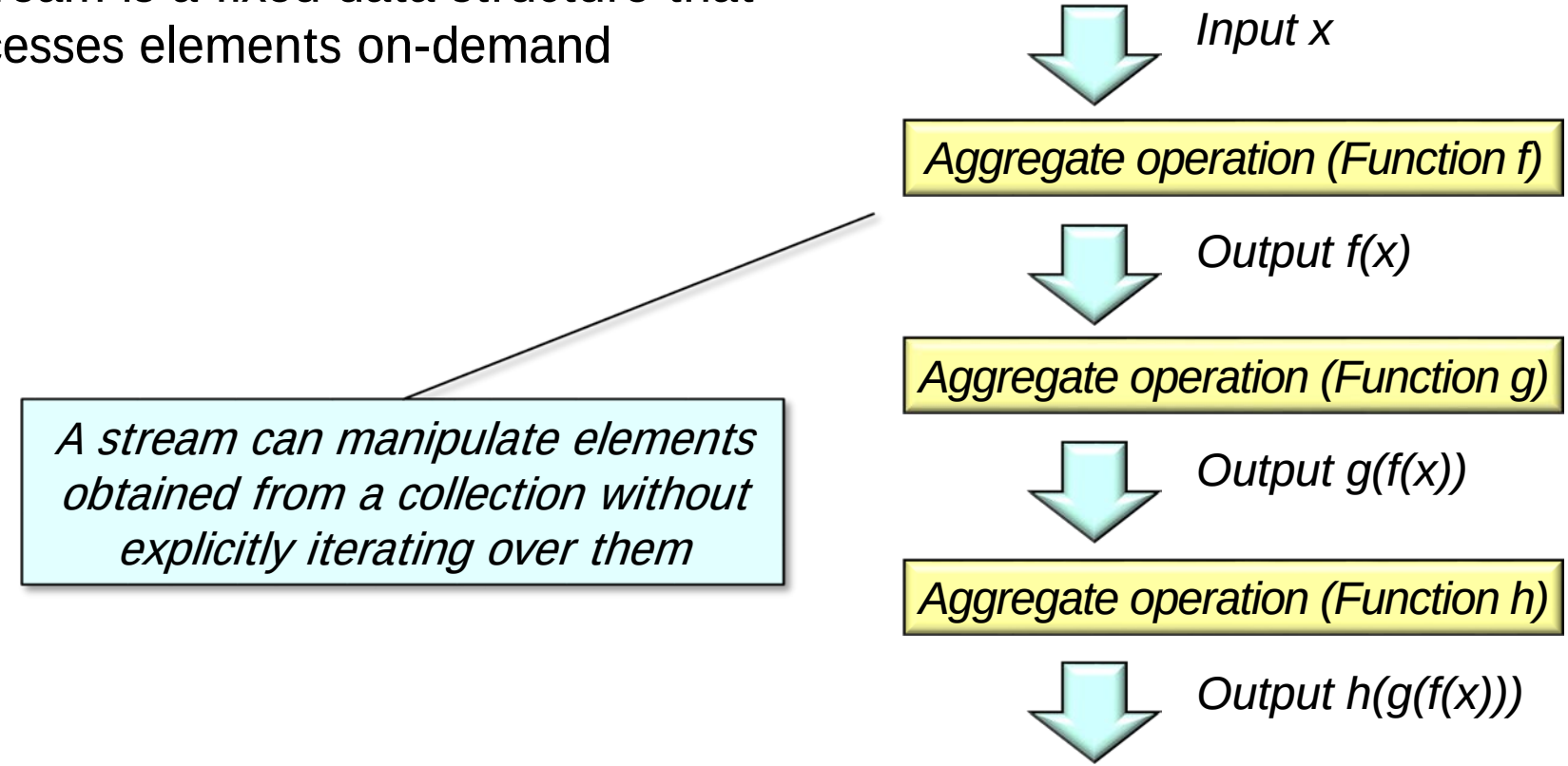
See docs.oracle.com/javase/tutorial/collections/intro

Contrasting Collections & Streams

- A collection is an in-memory data structure that can store, retrieve, & manipulate groups of elements
  - It is analogous to a DVD

Contrasting Collections & Streams

- A stream is a fixed data structure that processes elements on-demand



See tutorials.jenkov.com/java-collections/streams.html

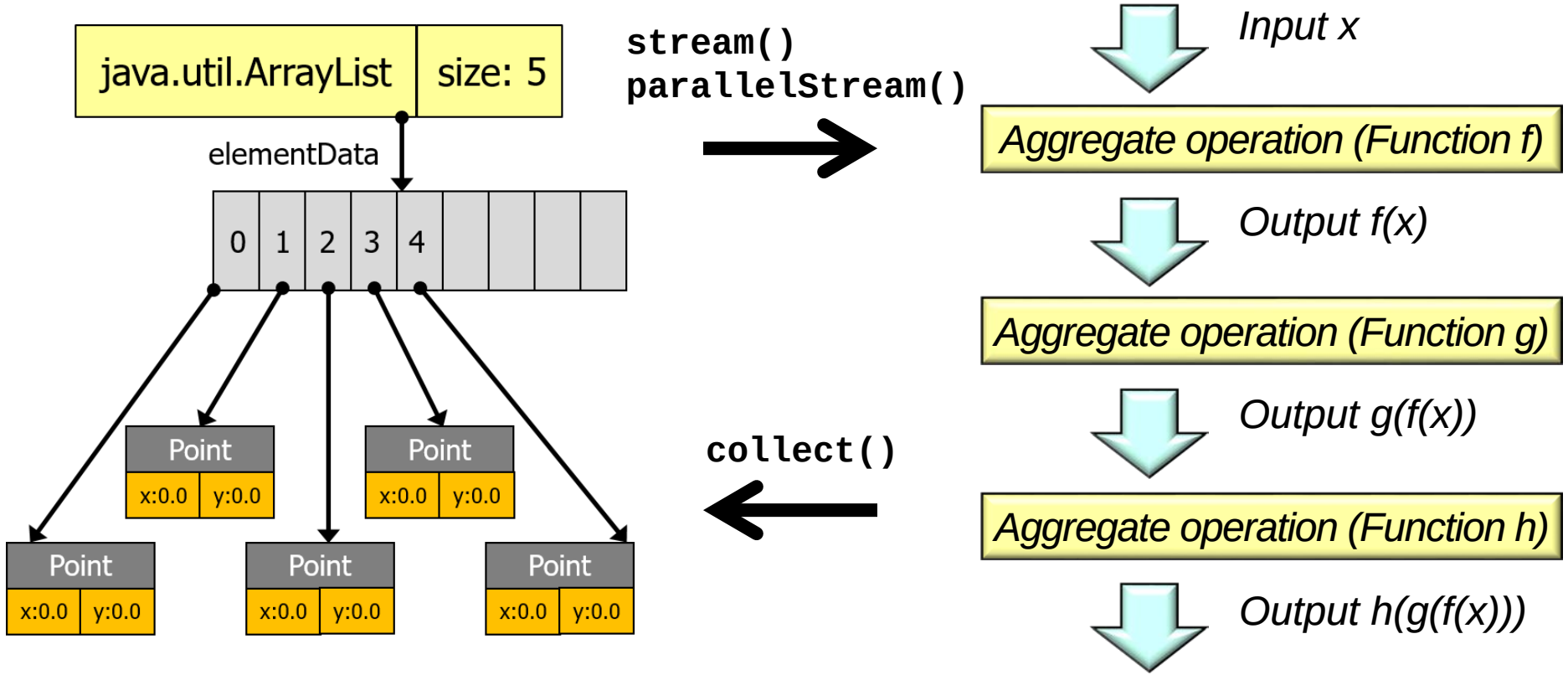
Contrasting Collections & Streams

- A stream is a fixed data structure that processes elements on-demand
  - A Java stream is analogous to a flow of bytes in a streaming video

Content is dynamically received & processed

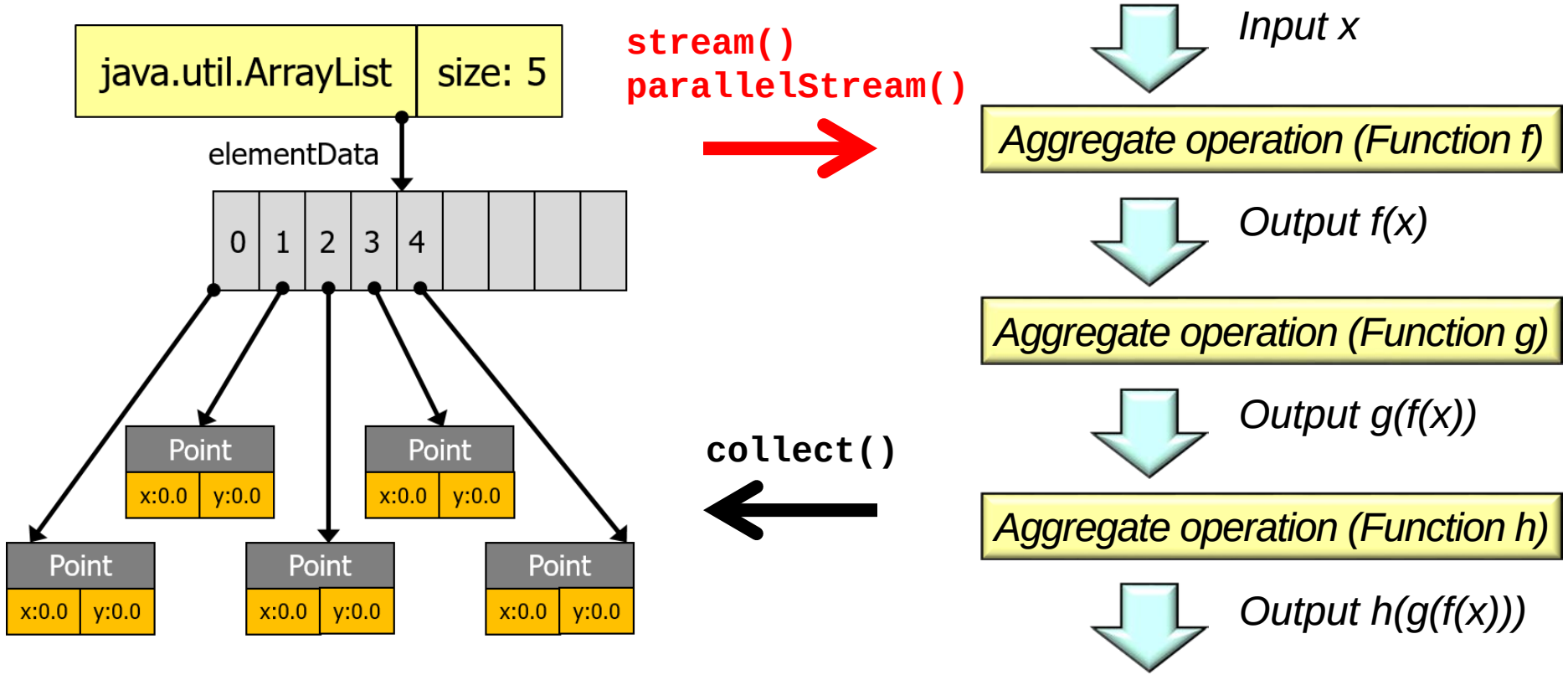
Contrasting Collections & Streams

- Various factory methods can convert collections to streams & vice versa



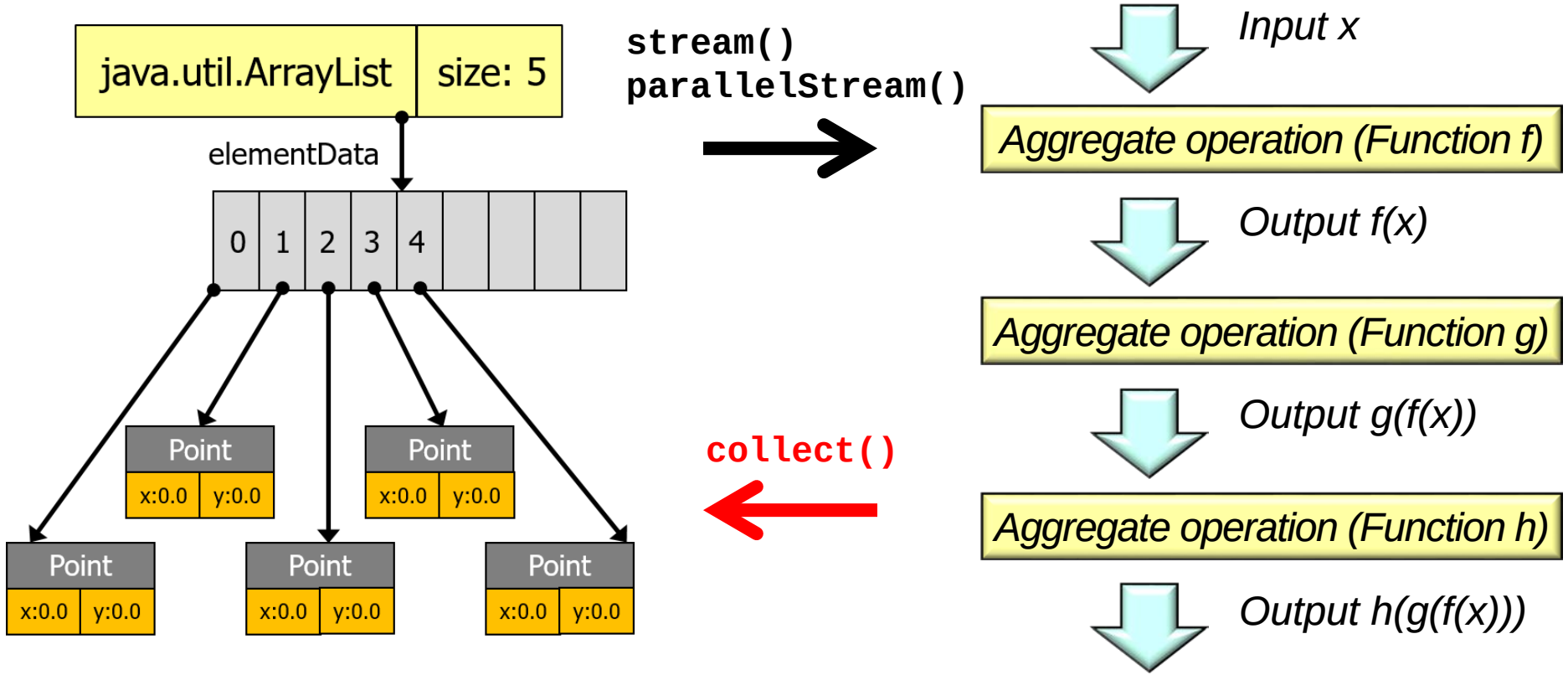
Contrasting Collections & Streams

- Various factory methods can convert collections to streams & vice versa



Contrasting Collections & Streams

- Various factory methods can convert collections to streams & vice versa



Contrasting Collections & Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

List<String> urls = Arrays.asList(urlArray);



for (int i = 0; i < urls.size(); ++i)
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
```

Contrasting Collections & Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

List<String> urls = Arrays.asList(urlArray);



for (int i = 0; i < urls.size(); ++i)
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
```

Contrasting Collections & Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

List<String> urls = Arrays.asList(urlArray);
```

Explicitly iterate through a list & modify each value

```
for (int i = 0; i < urls.size(); ++i)
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
```

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<String> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<String> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

Implicitly iterate through a pipeline of elements from a collection source, transform each value, & create a collection result

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<String> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

Implicitly iterate through a pipeline of elements from a collection source, transform each value, & create a collection result

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<String> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

Implicitly iterate through a pipeline of elements from a collection source, transform each value, & create a collection result

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

List<String> urls = Stream
  .of(urlArray)
  .map(s ->
       s.replace("cse.wustl", "dre.vanderbilt"))
  .collect(toList());
```

Implicitly iterate through a pipeline of elements from a collection source, transform each value, & create a collection result

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<String> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

Like iterators, elements in a stream can only be visited once during its lifetime

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<URL> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .map(rethrowFunction(URL::new))
   .collect(toList());
```

Key to the power of Java 8 streams is their ability to chain transformations

Contrasting Collections & Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu.edu/~schmidt/ka.png",
  "http://www.cse.wustl.edu.edu/~schmidt/robot.png",
  "http://www.cse.wustl.edu.edu/~schmidt/kitten.png"};

 List<URL> urls = Stream
   .of(urlArray)
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .map(rethrowFunction(URL::new))
   .collect(toList());
```

rethrowFunction() converts checked exception into runtime exception

See stackoverflow.com/a/27661504/3312330

End of Contrasting Java 8 Streams with Other Java Libraries