Java Streams: Common Operations

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

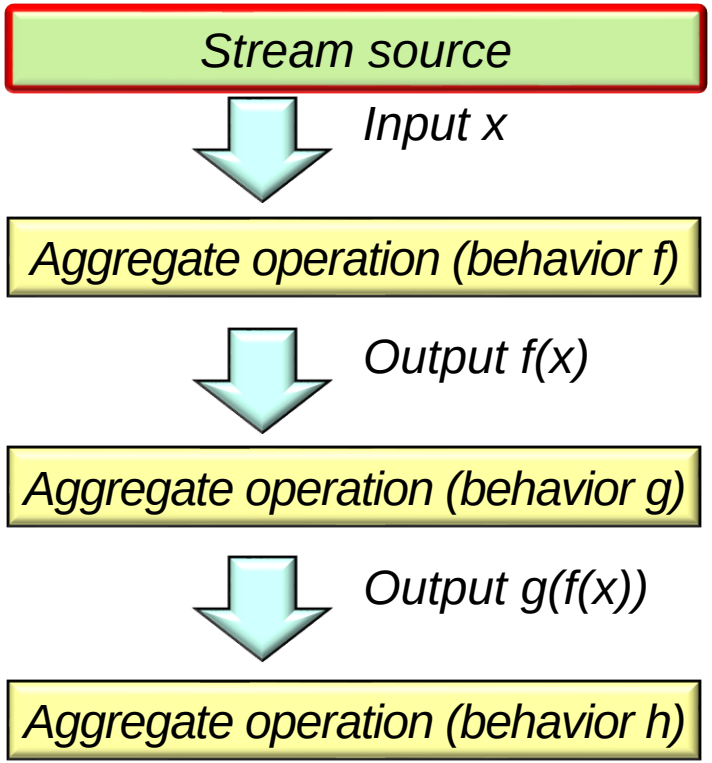
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

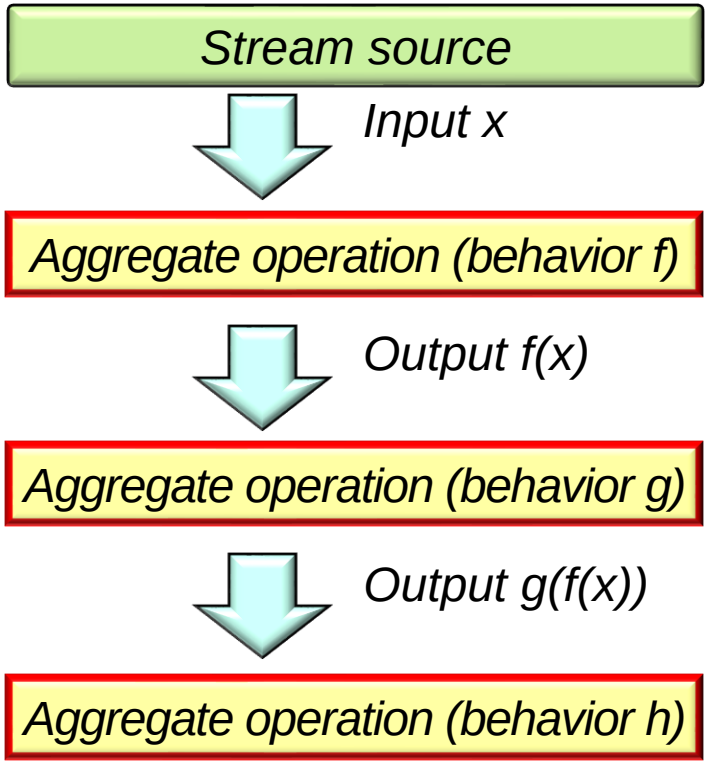
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java streams, e.g.,
  - Fundamentals of streams
  - Benefits of streams
  - Operations that create a stream



Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java streams, e.g.,
  - Fundamentals of streams
  - Benefits of streams
  - Operations that create a stream
  - Aggregate operations in a stream

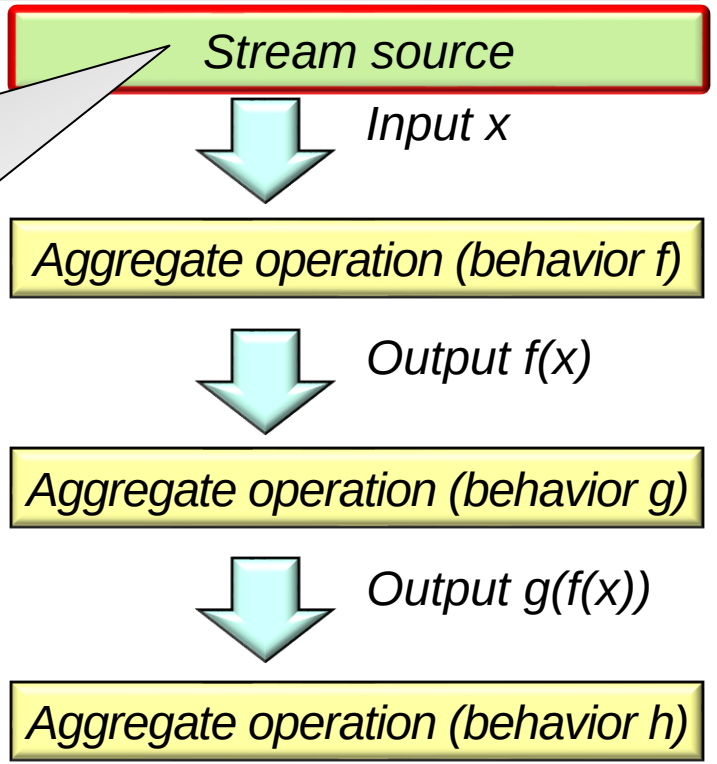


Operations that Create a Java Stream

Operations that Create a Java Stream

- A factory method creates a stream from some source

```
Stream
   .of("horatio",
       "laertes",
       "Hamlet", ...)
   ...
```

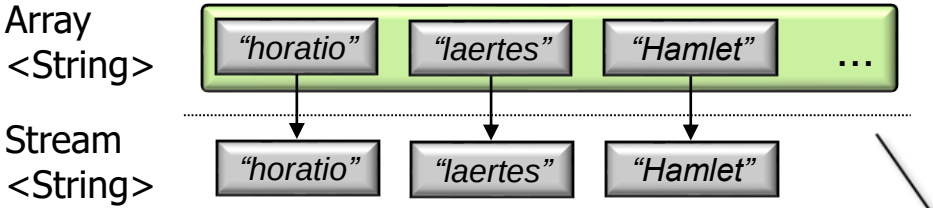


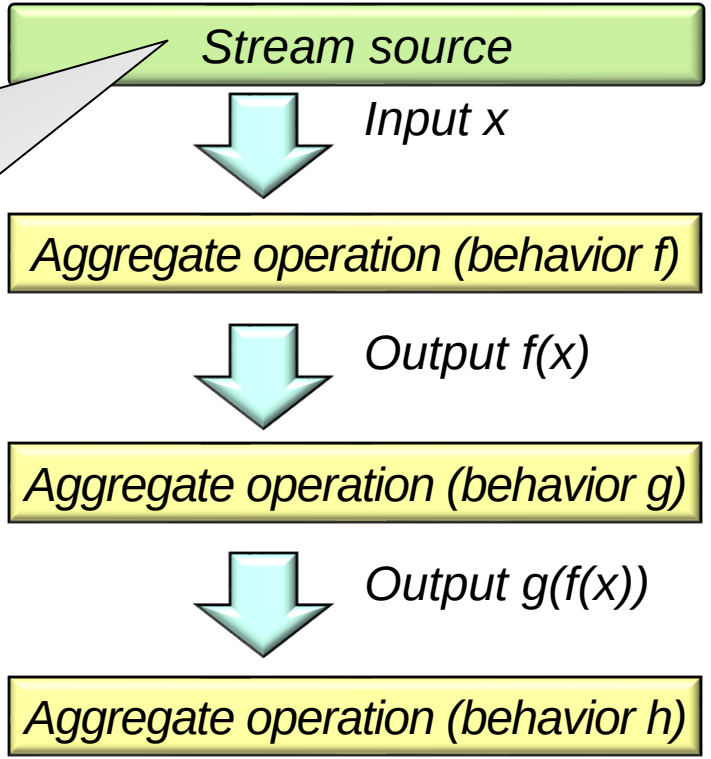
See en.wikipedia.org/wiki/Factory_method_pattern

Operations that Create a Java Stream

- A factory method creates a stream from some source

```
Stream
   .of("horatio",
       "laertes",
       "Hamlet", ...) ...
```





The of() factory method converts an array of T into a stream of T

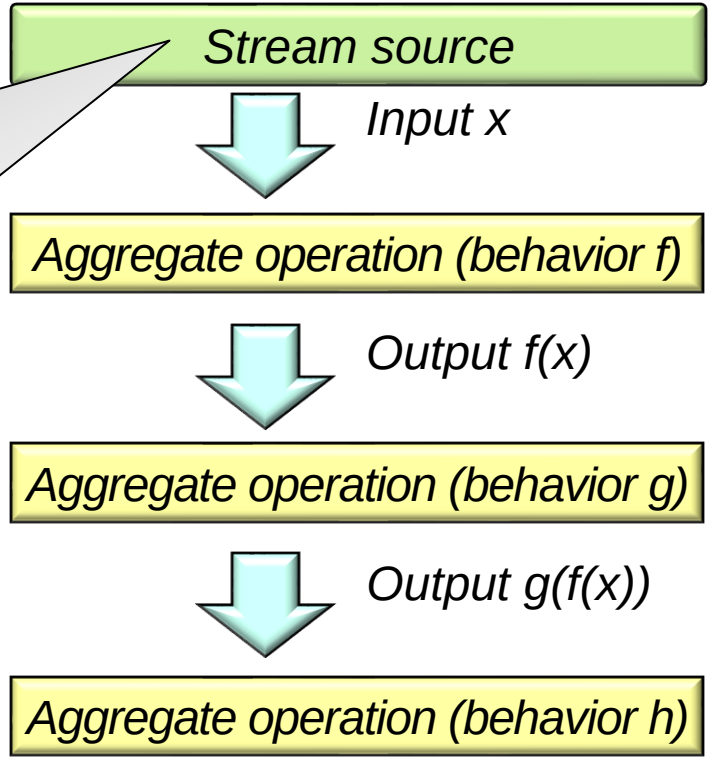
See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of

Operations that Create a Java Stream

- A factory method creates a stream from some source

```
List<String> l1 = ...;
List<String> l2 = ...;
List<String> l3 = ...;

Stream
   .of(l1, l2, l3)
   .flatMap(List::stream)
   ...
   .forEach(System.out::println);
```



of() is flexible, especially when combined with other aggregate operations

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12

Operations that Create a Java Stream

- Many factory methods create streams

```
collection.stream()
collection.parallelStream()
Pattern.compile(…).splitAsStream()
Stream.of(value1,… ,valueN)
StreamSupport
   .stream(iterable.spliterator(),
           false)
...
```

```
Arrays.stream(array)
Arrays.stream(array, start, end)
Files.lines(file_path)
"string".chars()
Stream.iterate(init_value,
                  generate_expression)
Stream.builder().add(...).build()
Stream.generate(supplier)
Files.list(file_path)
Files.find(file_path, max_depth,
              matcher)
...
```

Operations that Create a Java Stream

- Many factory methods create streams

```
collection.stream()
collection.parallelStream()
Pattern.compile(…).splitAsStream()
Stream.of(value1,… ,valueN)
StreamSupport
   .stream(iterable.spliterator(),
           false)
...
```

```
Arrays.stream(array)
Arrays.stream(array, start, end)
Files.lines(file_path)
"string".chars()
Stream.iterate(init_value,
                  generate_expression)
Stream.builder().add(...).build()
Stream.generate(supplier)
Files.list(file_path)
Files.find(file_path, max_depth,
           matcher)
...
```

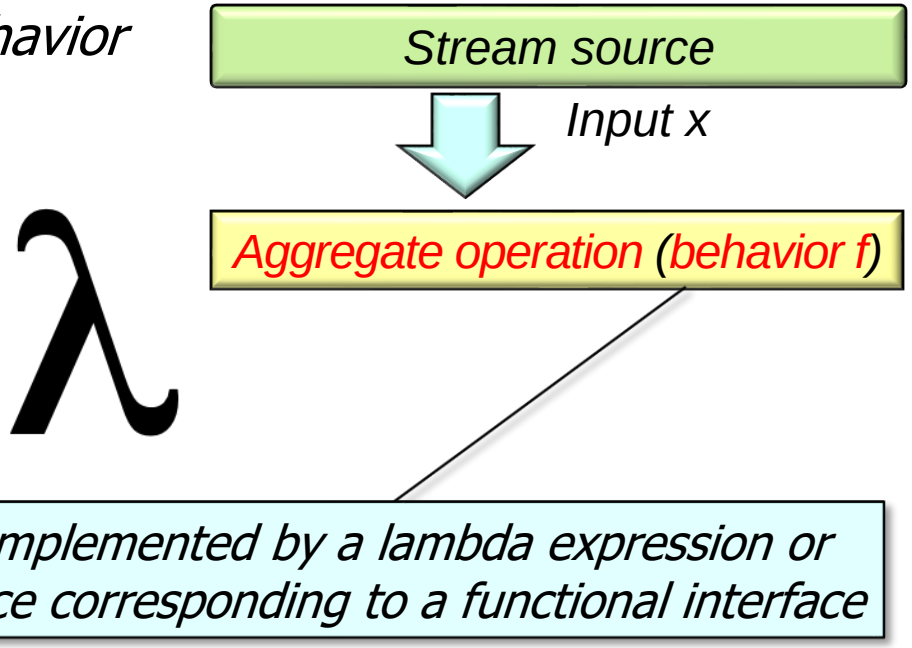
These are key factory methods that we focus on in this course.

See the upcoming lesson on "*Java Streams: Common Factory Methods*"

Java Streams
Aggregate Operations

Java Streams Aggregate Operations

- An aggregate operation performs a *behavior* on each element in a stream



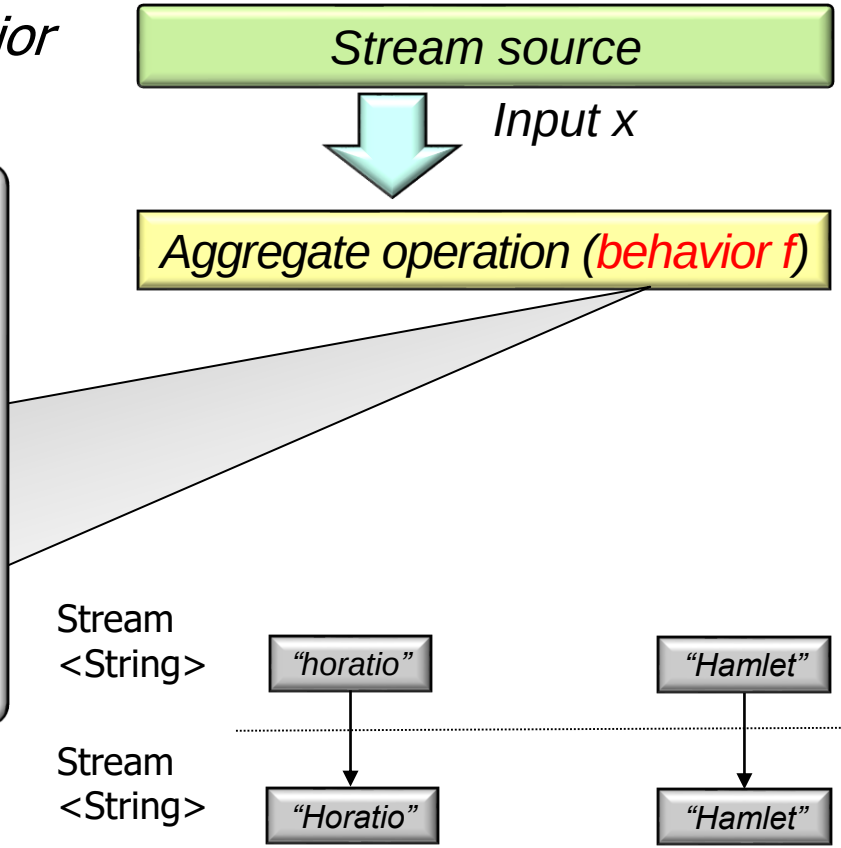
See blog.indrek.io/articles/java-8-behavior-parameterization

Java Streams Aggregate Operations

- An aggregate operation performs a *behavior* on each element in a stream

```
Stream
   .of("horatio",
       "laertes",
       "Hamlet", ...)
   .filter(s -> toLowerCase
               (s.charAt(0)) == 'h')
   .map(this::capitalize)
   .sorted()
   .forEach(System.out::println);
```

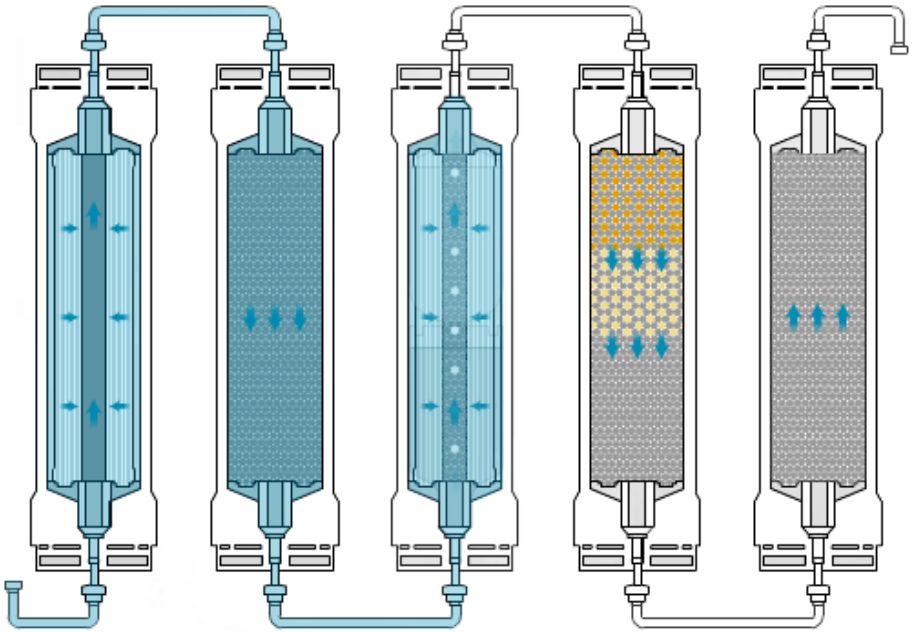
Method reference

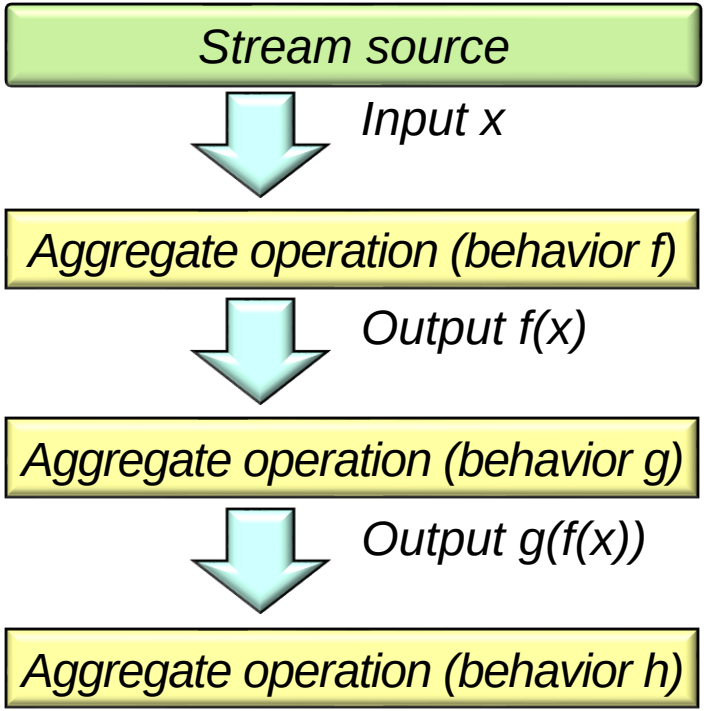


See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12

Java Streams Aggregate Operations

- Aggregate operations can be composed to form a pipeline of processing phases

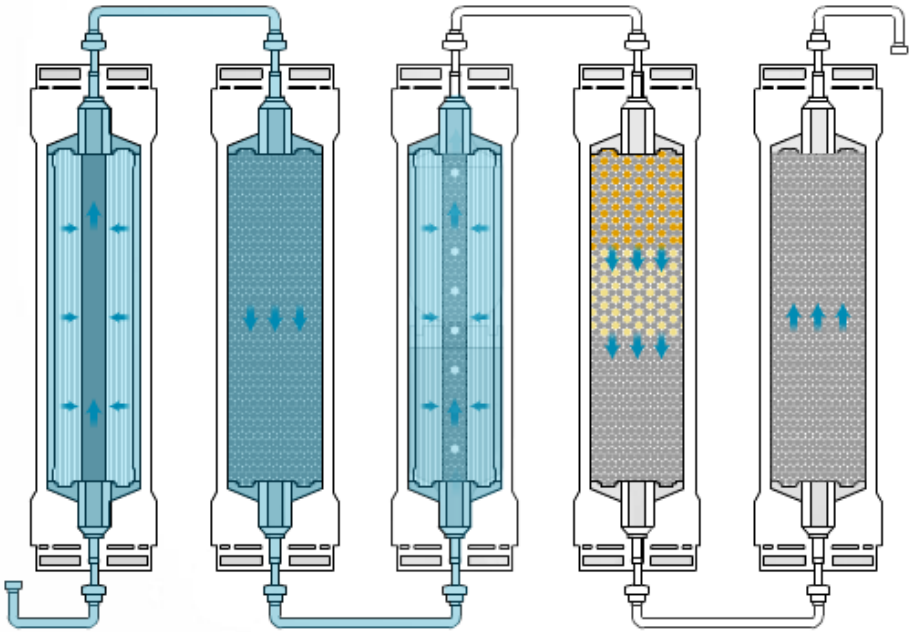


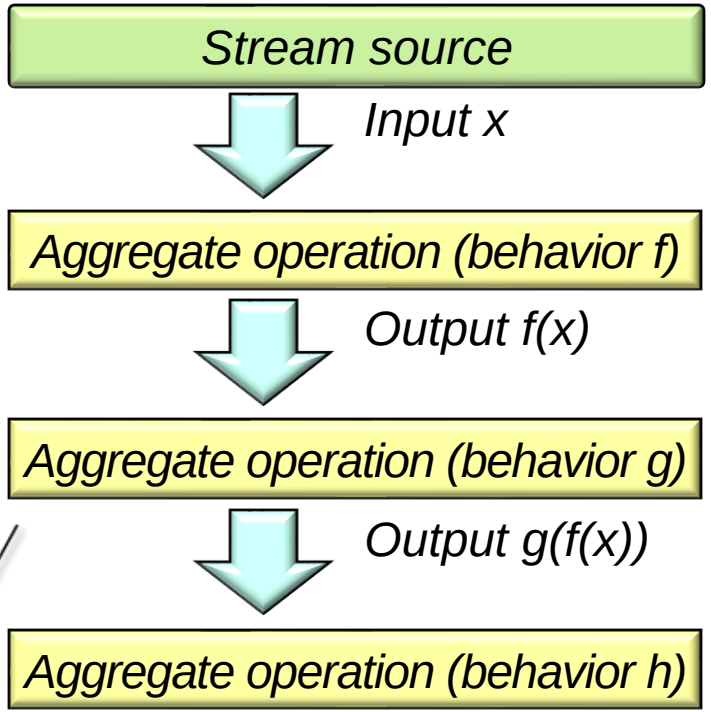


See en.wikipedia.org/wiki/Pipeline_(software)

Java Streams Aggregate Operations

- Aggregate operations can be composed to form a pipeline of processing phases



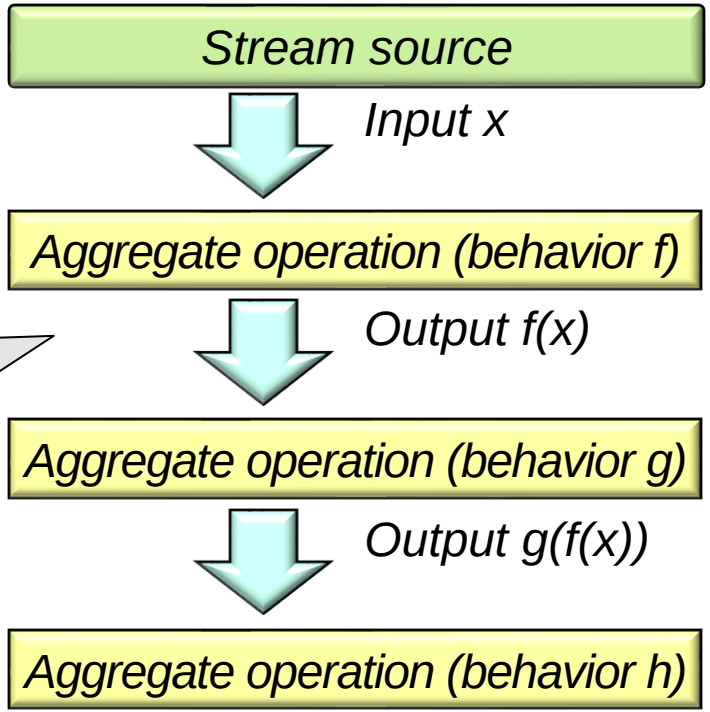


The output of one aggregate operation can be input into the next one in the stream.

Java Streams Aggregate Operations

- Aggregate operations can be composed to form a pipeline of processing phases

```
Stream
   .of("horatio",
       "laertes",
       "Hamlet", ...)
   .filter(s -> toLowerCase
            (s.charAt(0)) == 'h')
   .map(this::capitalize)
   .sorted()
   .forEach(System.out::println);
```

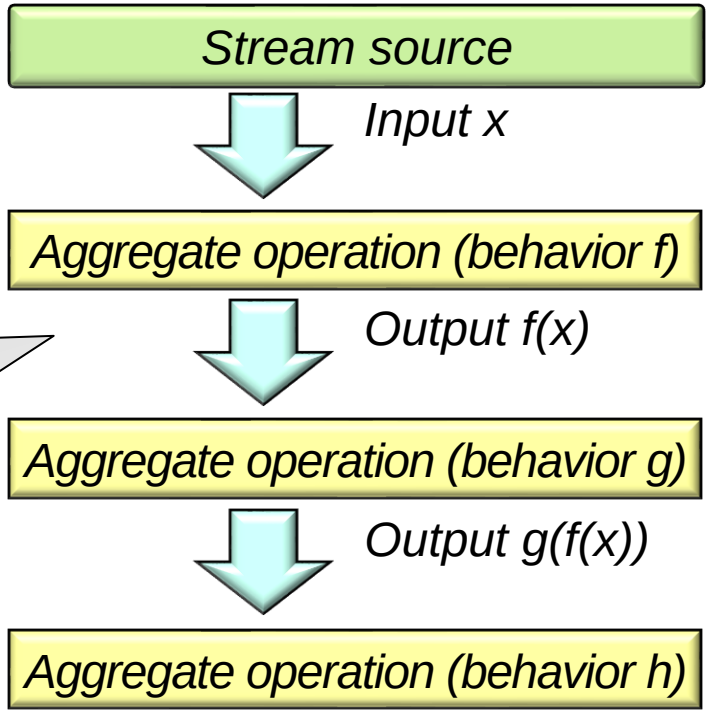


The output of one aggregate operation can be input into the next one in the stream.

Java Streams Aggregate Operations

- Aggregate operations can be composed to form a pipeline of processing phases

```
Stream
   .of("horatio",
       "laertes",
       "Hamlet", ...)
   .filter(s -> toLowerCase
            (s.charAt(0)) == 'h')
   .map(this::capitalize)
   .sorted()
   .forEach(System.out::println);
```



Aggregate operations iterate internally (& invisibly) unlike collections, which are iterated explicitly using an iterator.

See www.javabrahman.com/java-8/java-8-internal-iterators-vs-external-iterators

End of Java Streams: Common Operations