Java Streams Internals: Splitting & Combining

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

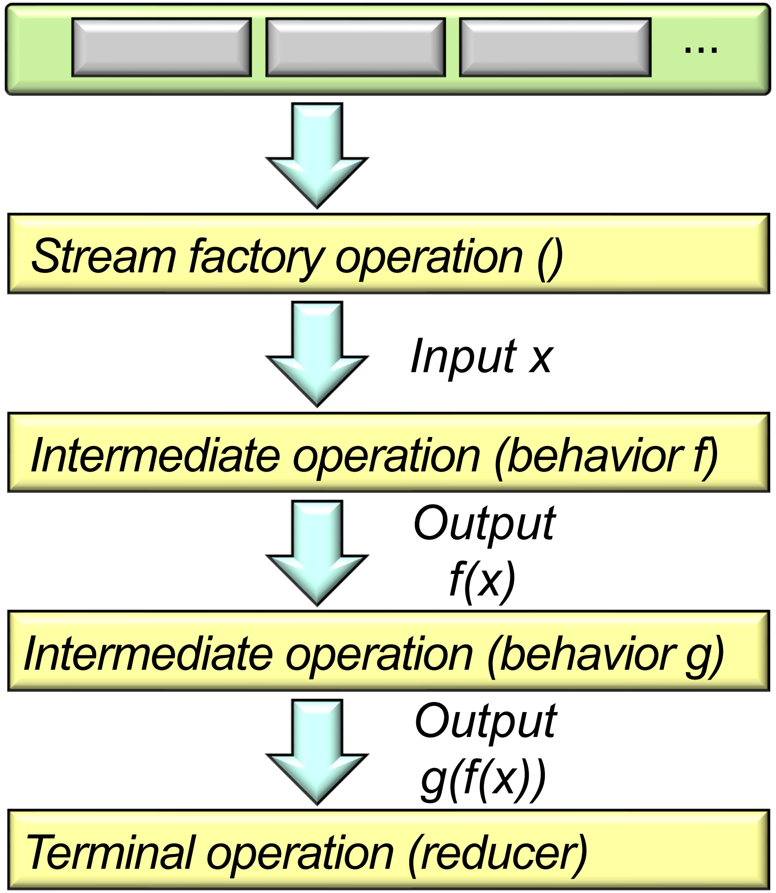
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand stream internals


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

See developer.ibm.com/technologies/java/articles/j-java-streams-3-brian-goetz

Learning Objectives in this Part of the Lesson

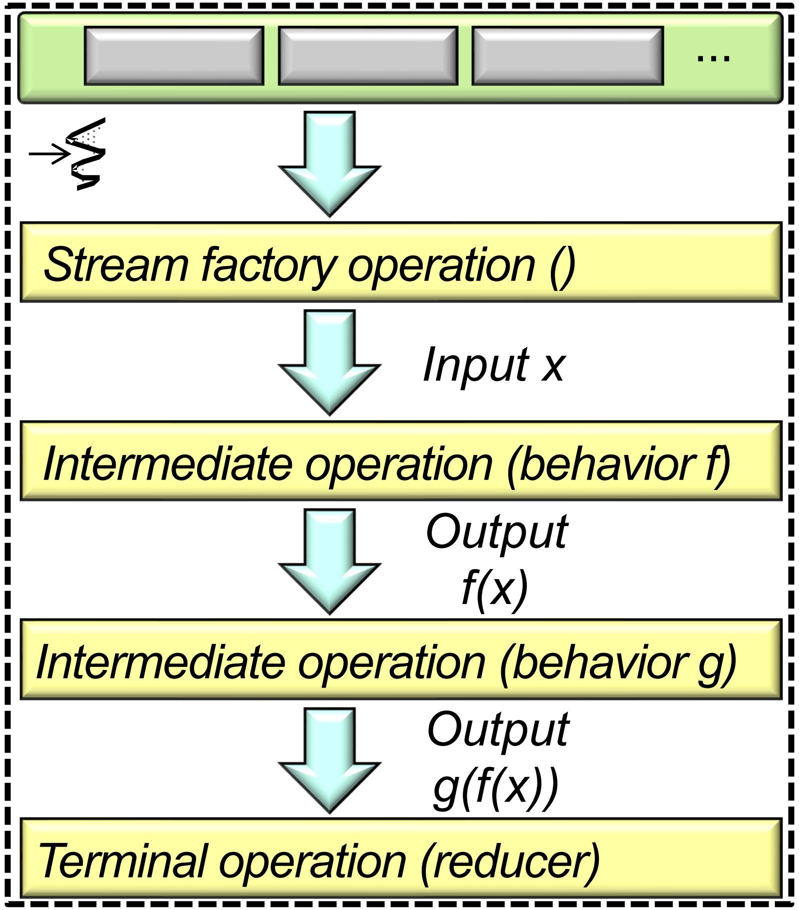
- Understand stream internals, e.g.
  - Know what can change & what can't

See en.wikipedia.org/wiki/Serenity_Prayer

Why Knowledge of Streams Internals Matters

Why Knowledge of Streams Internals Matters

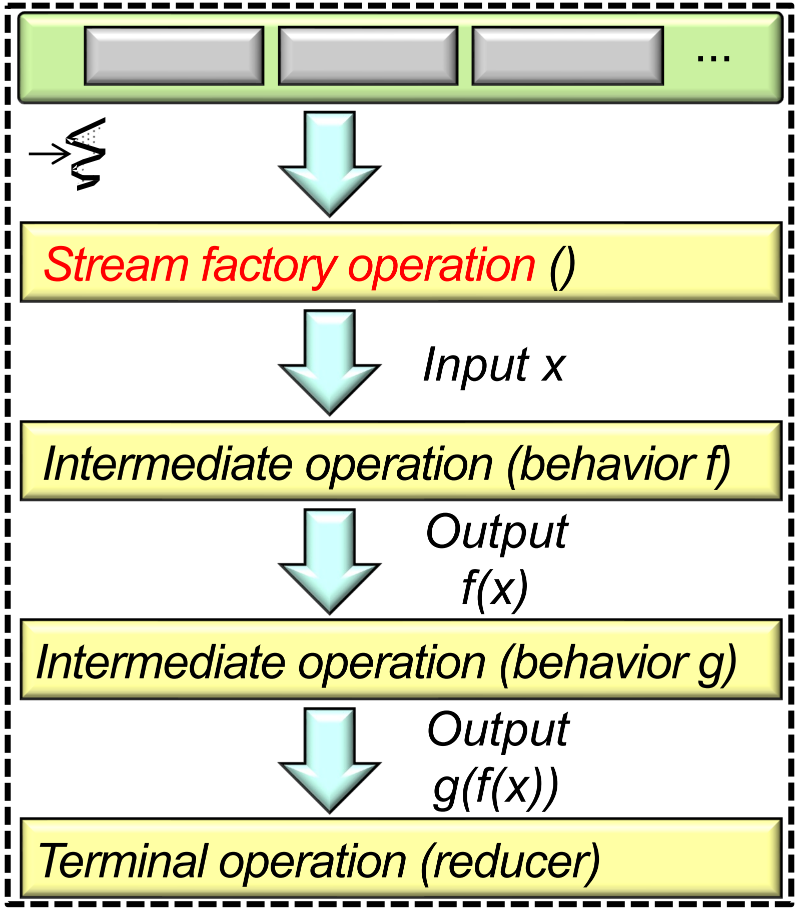
- A Java stream consists of three phases


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

See www.jstatsoft.org/article/view/v040i01/v40i01.pdf

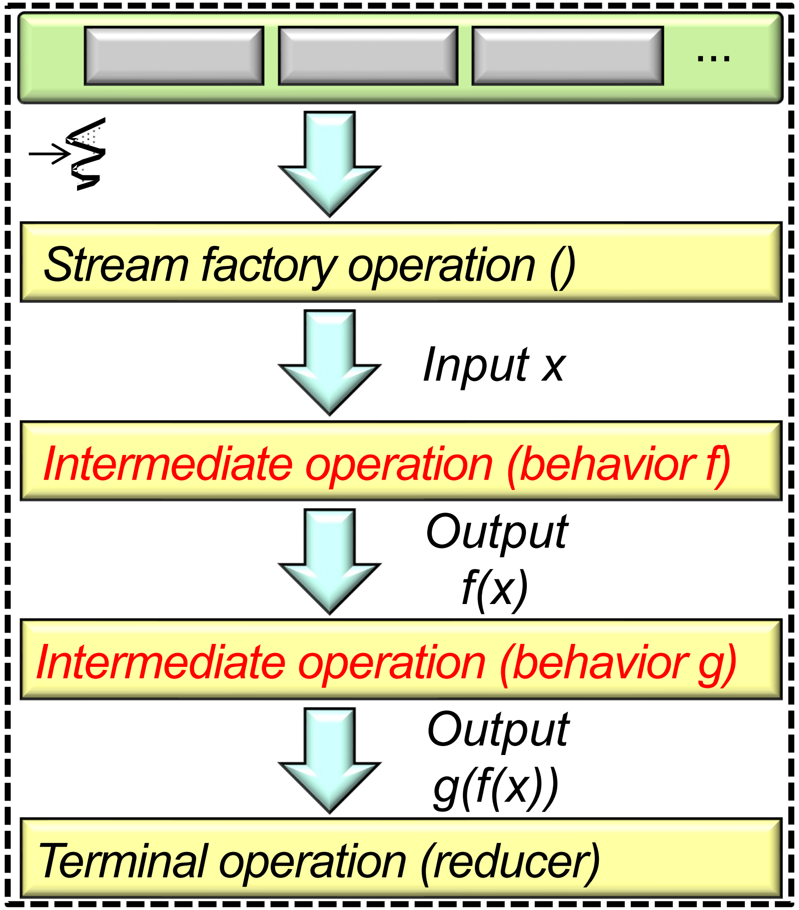
Why Knowledge of Streams Internals Matters

- A Java stream consists of three phases
  - *Split* – Uses a spliterator to convert a data source into a stream


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

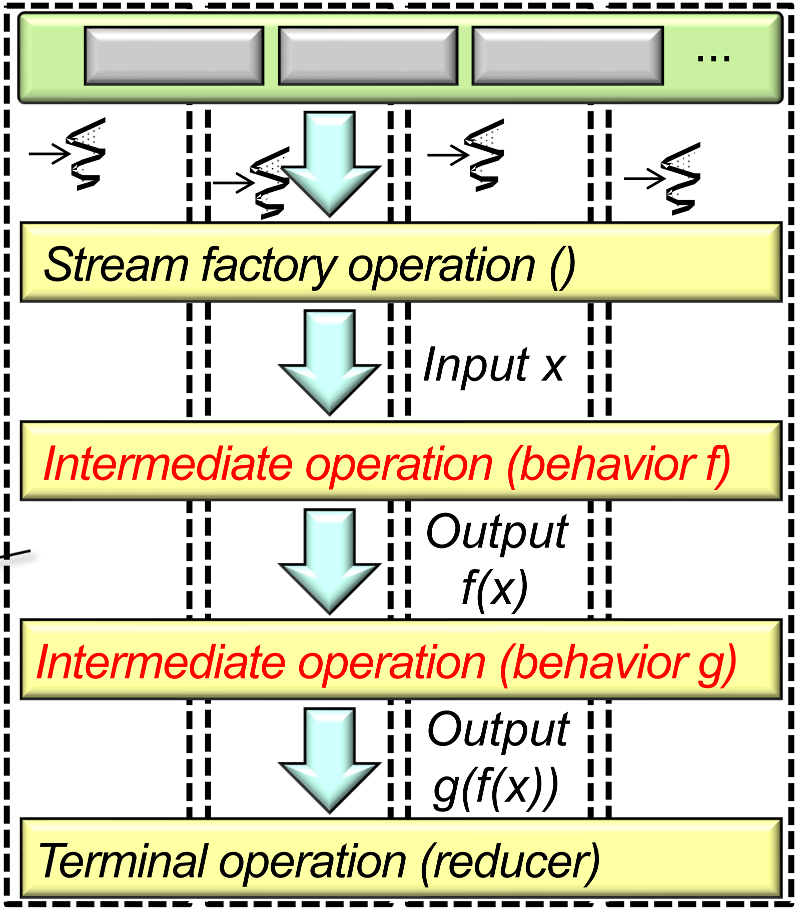
Why Knowledge of Streams Internals Matters

- A Java stream consists of three phases
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

Why Knowledge of Streams Internals Matters

- A Java stream consists of three phases
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream

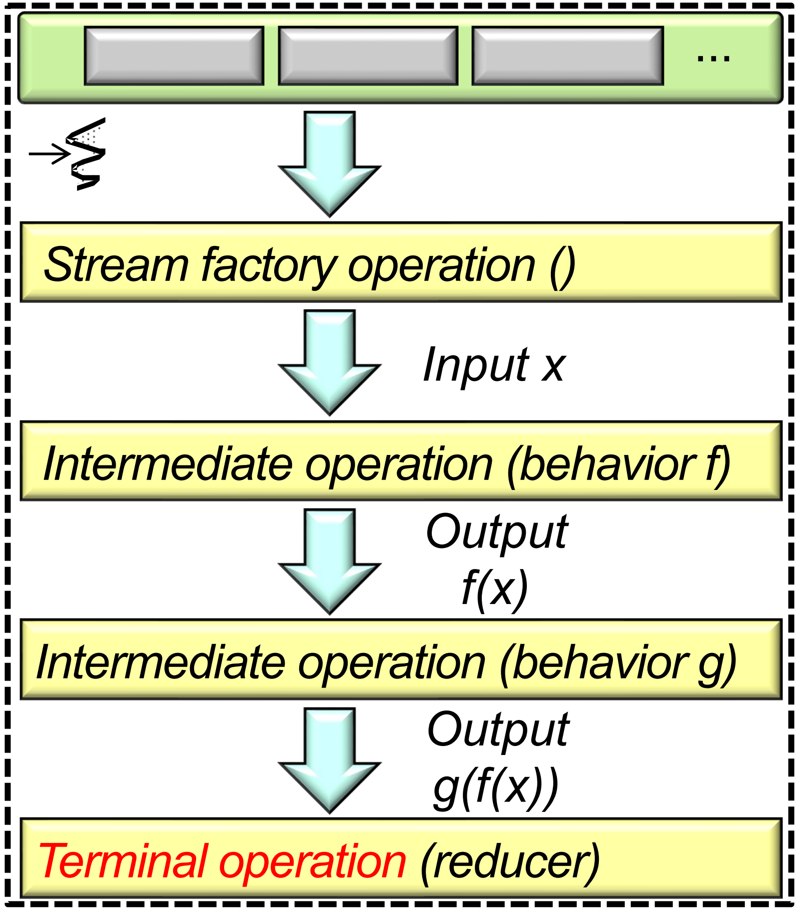

...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

A parallel stream can process these elements simultaneously.

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

Why Knowledge of Streams Internals Matters

- A Java stream consists of three phases
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream
  - *Combine* – Trigger intermediate operation processing & create a single result


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

Why Knowledge of Streams Internals Matters

- A Java stream consists of three phases
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream
  - *Combine* – Trigger intermediate operation processing & create a single result

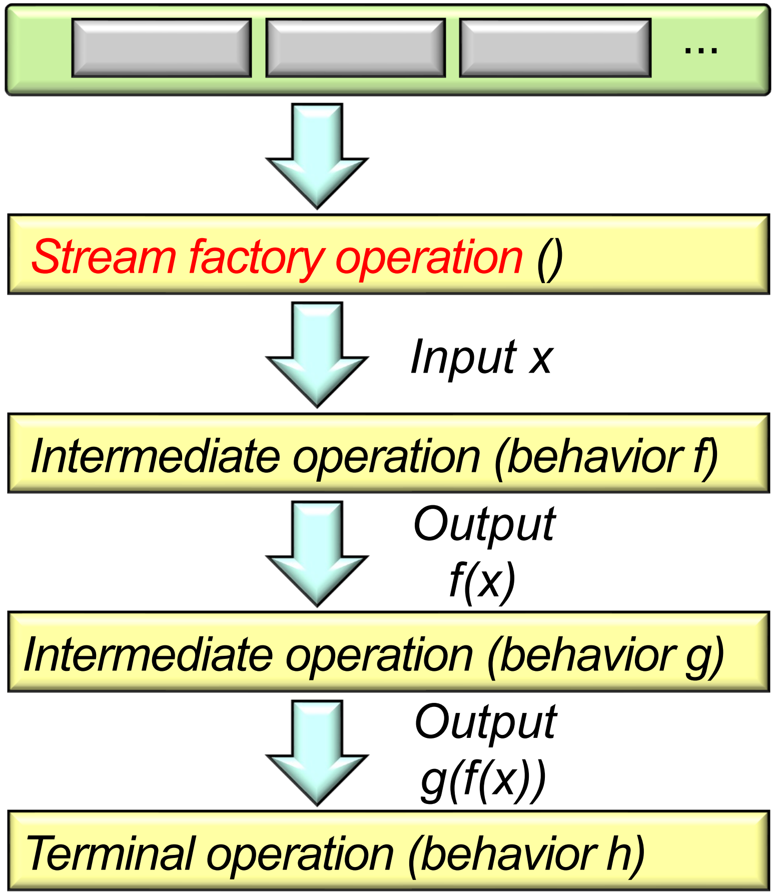
Knowing which of these three phases you can control (& how to control them) is important!

We focus on sequential stream internals now & parallel stream internals later

Java Streams Splitting & Combining Mechanisms

Java Streams Splitting & Combining Mechanisms

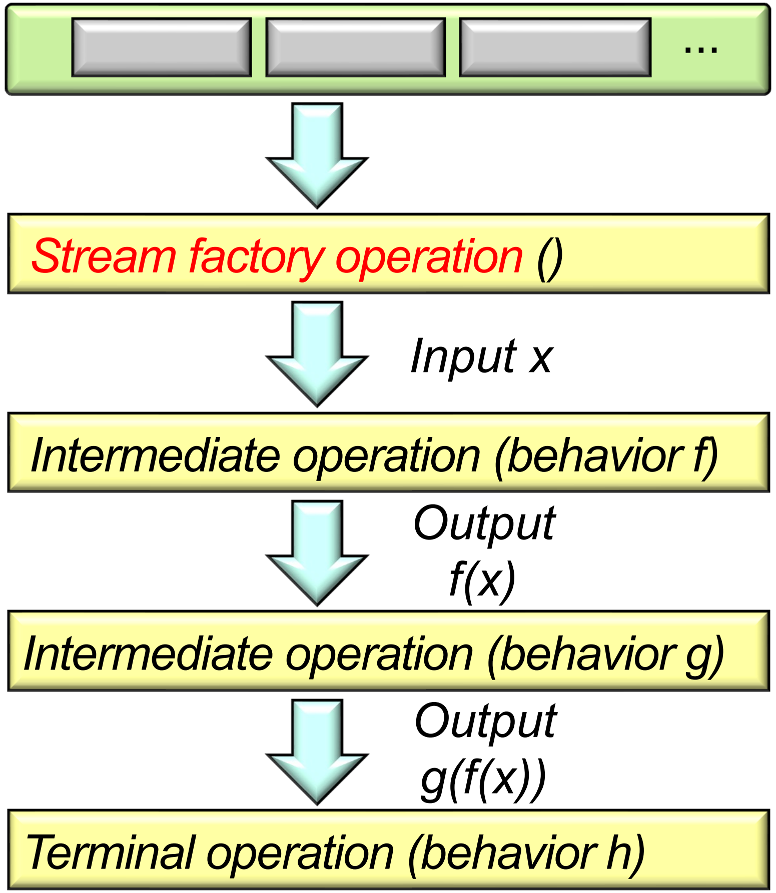
- A stream's splitting & combining mechanisms are often invisible


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

Java Streams Splitting & Combining Mechanisms

- A stream's splitting & combining mechanisms are often invisible, e.g.
  - All Java collections have predefined spliterators

```
interface Collection<E> {
   ...
   default Spliterator<E> spliterator() {
     return Spliterators
       .spliterator(this, 0);
   }

   default Stream<E> stream() {
     return StreamSupport
       .stream(spliterator(), false);
   }
  ...
}
```

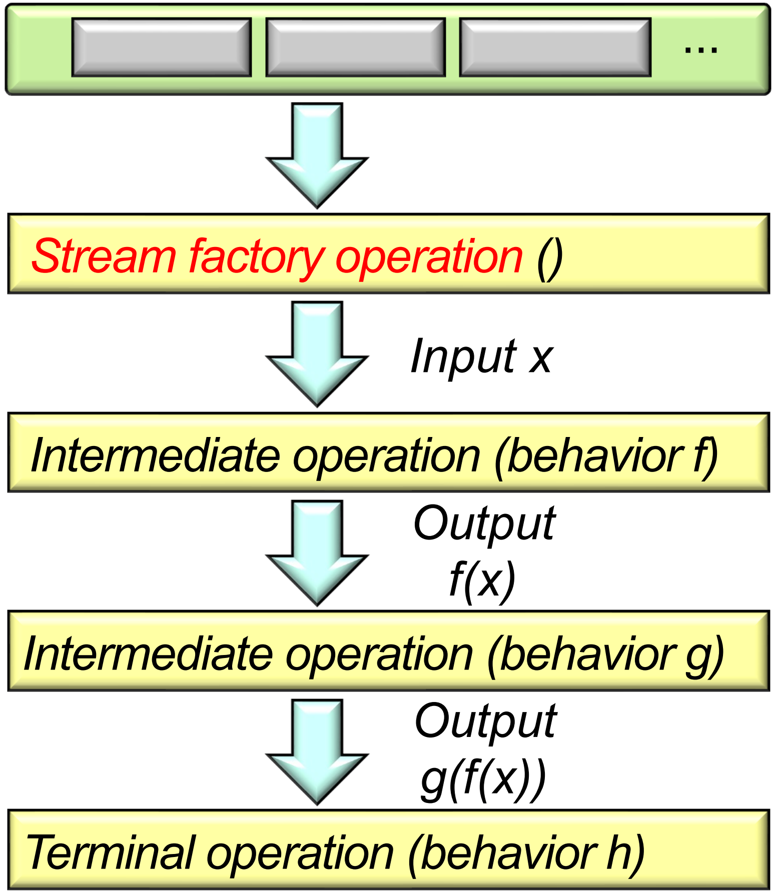

...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

See docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Java Streams Splitting & Combining Mechanisms

- A stream's splitting & combining mechanisms are often invisible, e.g.
  - All Java collections have predefined spliterators

```
interface Collection<E> {
  ...
  default Spliterator<E> spliterator() {
    return Spliterators
      .spliterator(this, 0);
  }

  default Stream<E> stream() {
    return StreamSupport
      .stream(spliterator(), false);
  }
 ...
}
```

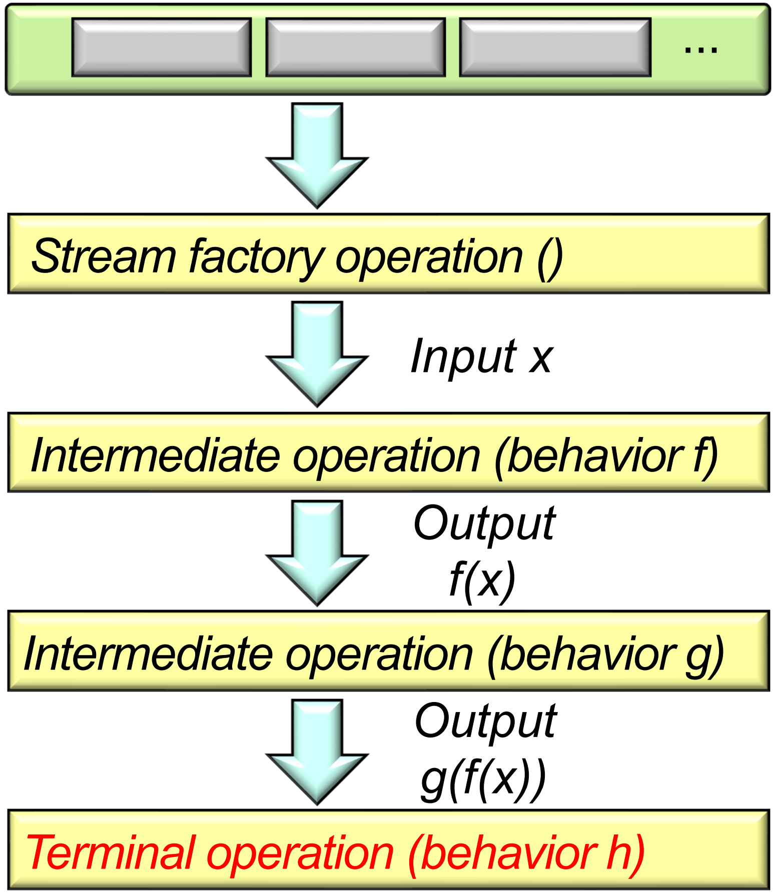

...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html

Java Streams Splitting & Combining Mechanisms

- A stream's splitting & combining mechanisms are often invisible, e.g.
  - All Java collections have predefined spliterators
  - Java also predefines collector factory methods in the Collectors utility class

```
final class Collectors {
  ...
  public static <T> Collector<T, ?, List<T>>
    toList() { ... }

  public static <T> Collector<T, ?, Set<T>>
    toSet() { ... }
  ...
}
```

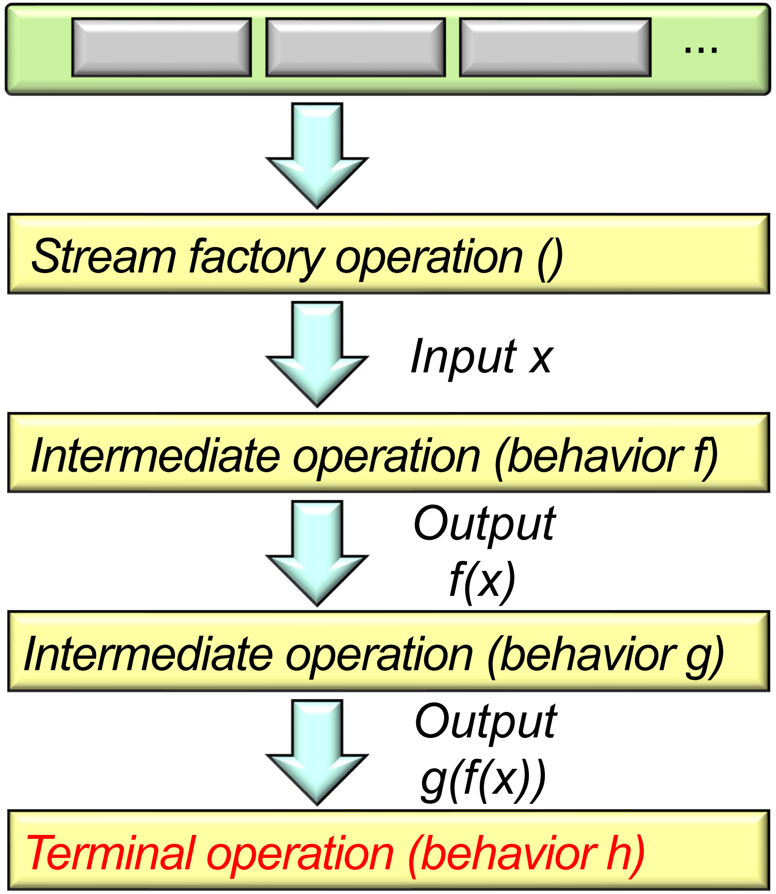

...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html

Java Streams Splitting & Combining Mechanisms

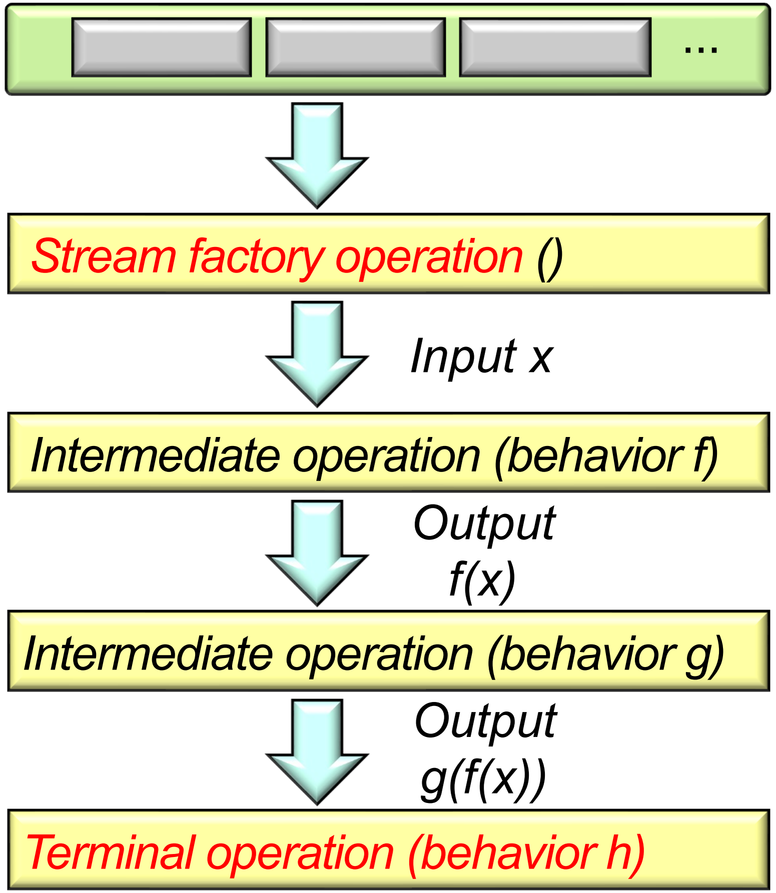
- A stream's splitting & combining mechanisms are often invisible, e.g.
  - All Java collections have predefined spliterators
  - Java also predefines collector factory methods in the Collectors utility class

```
final class Collectors {
  ...
  public static <T> Collector<T, ?, List<T>>
    toList() { ... }

  public static <T> Collector<T, ?, Set<T>>
    toSet() { ... }
  ...
}
```


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

Java Streams Splitting & Combining Mechanisms

- However, programmers can customize the behavior of splitting & combining

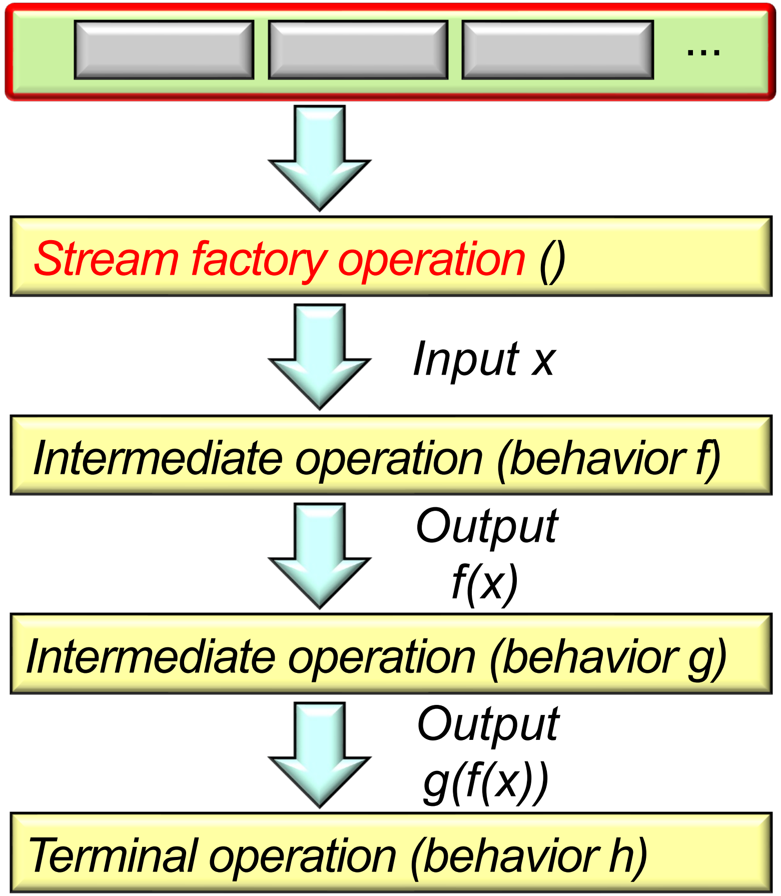

...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

Java Streams Splitting & Combining Mechanisms

- However, programmers can customize the behavior of splitting & combining

```
interface Spliterator<T> {
  boolean tryAdvance
     (Consumer<? Super T> action);

  Spliterator<T> trySplit();

  void forEachRemaining
     (Consumer<? Super T> action);

  long estimateSize();

  int characteristics();

}
```

An interface used to traverse & partition elements of a source.


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

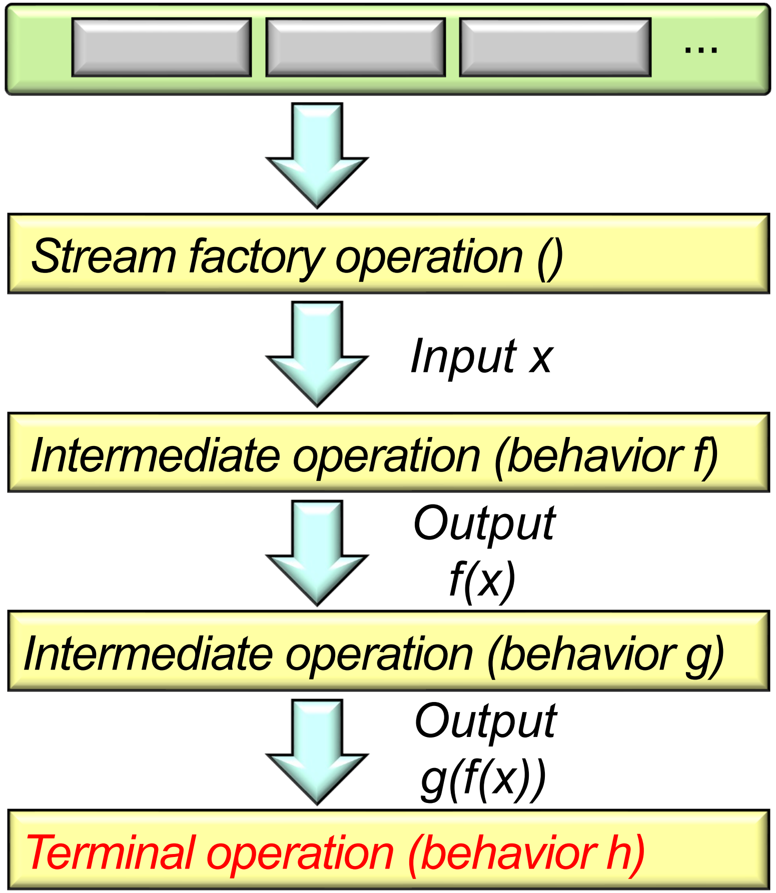
See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html

Java Streams Splitting & Combining Mechanisms

- However, programmers can customize the behavior of splitting & combining

```
interface Collector<T,A,R> {
  Supplier<A> supplier();
  BiConsumer<A, T> accumulator();
  BinaryOperator<A> combiner();
  Function<A, R> finisher();
  Set<Collector.Characteristics>
    characteristics()
  ...
}
```

A framework that accumulates input elements into a mutable result container.


...
Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collector.html

End of Java Streams Internals: Splitting & Combining