Overview of Java Streams Internals (Part 1)

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

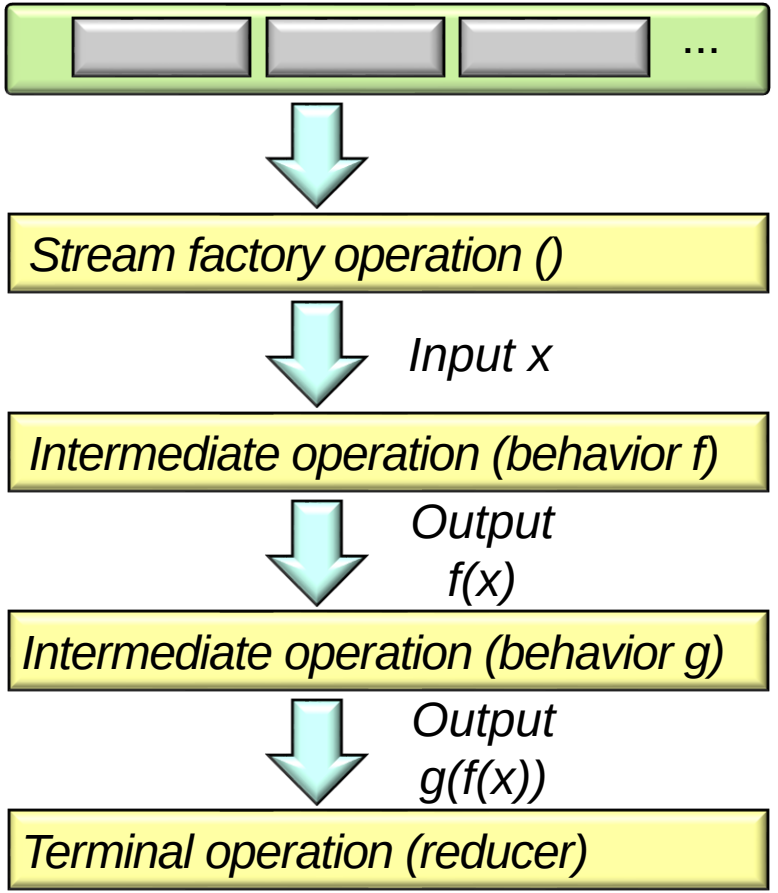
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand stream internals



See www.ibm.com/developerworks/library/j-java-streams-3-brian-goetz

Learning Objectives in this Part of the Lesson

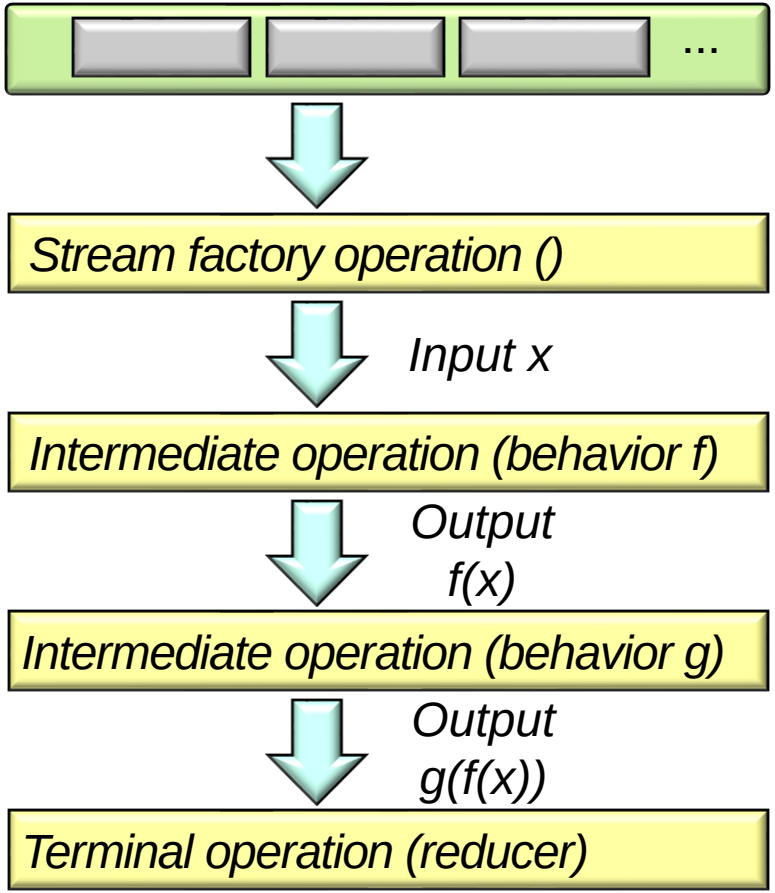
- Understand stream internals, e.g.
  - Know what can change & what can't

See en.wikipedia.org/wiki/Serenity_Prayer

Why Knowledge of Streams Internals Matters

Why Knowledge of Streams Internals Matters

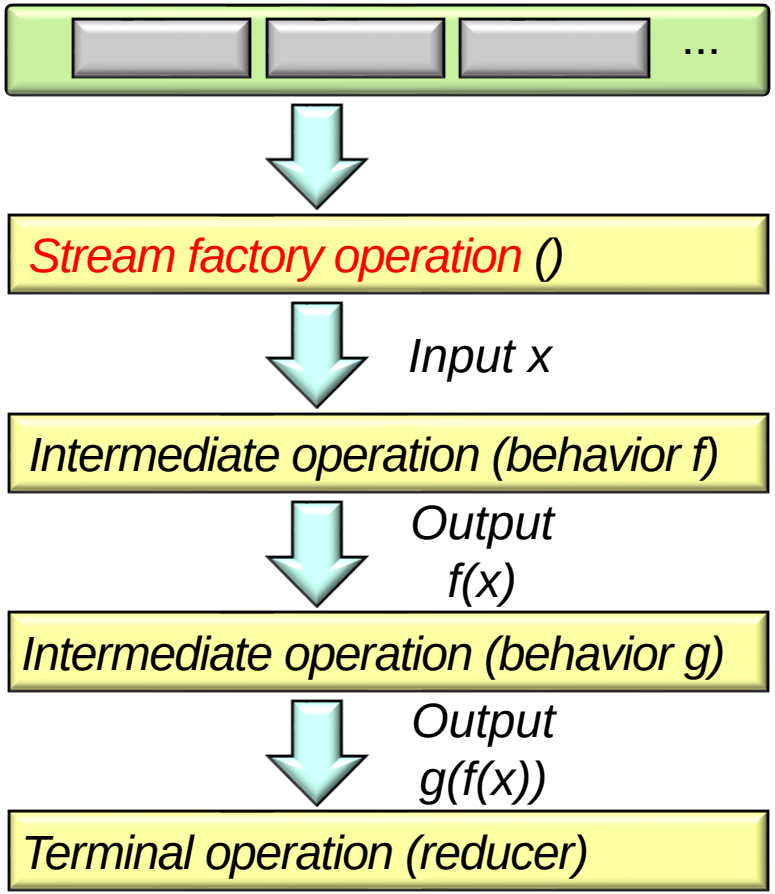
- Recall the 3 phases of a Java stream



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

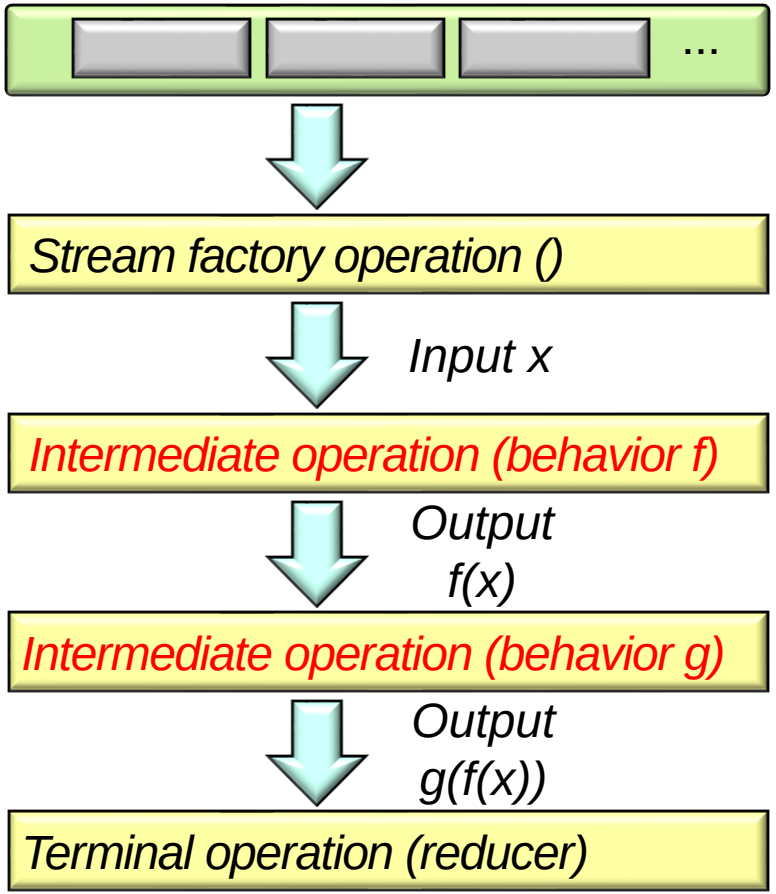
Why Knowledge of Streams Internals Matters

- Recall the 3 phases of a Java stream
  - *Split* – Uses a spliterator to convert a data source into a stream



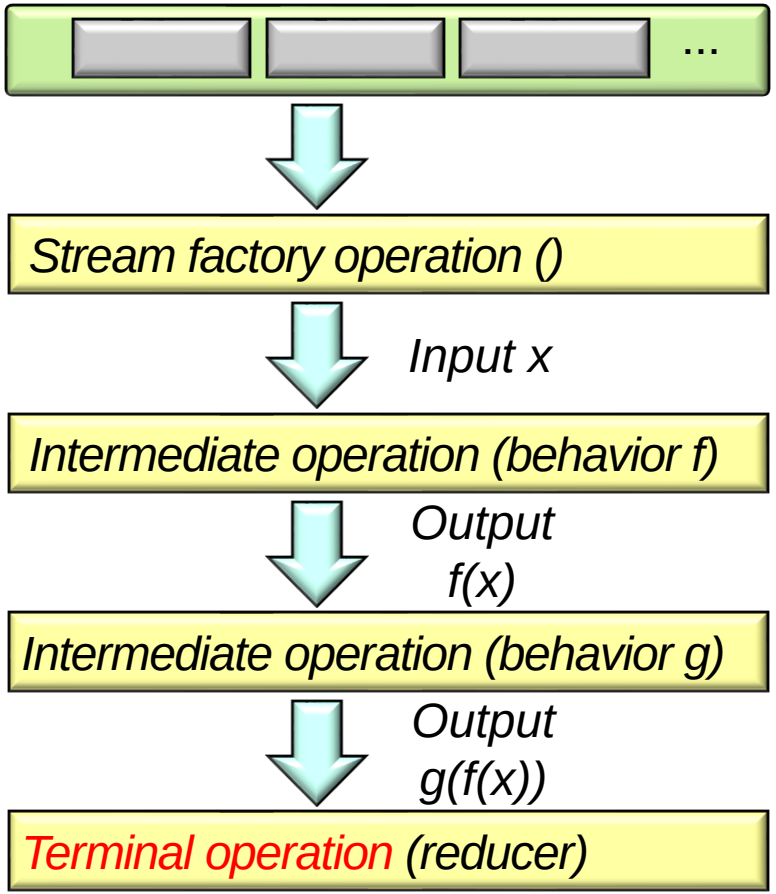
Why Knowledge of Streams Internals Matters

- Recall the 3 phases of a Java stream
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream



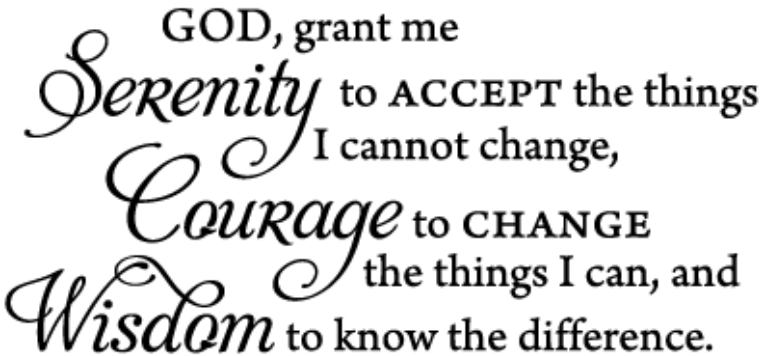
Why Knowledge of Streams Internals Matters

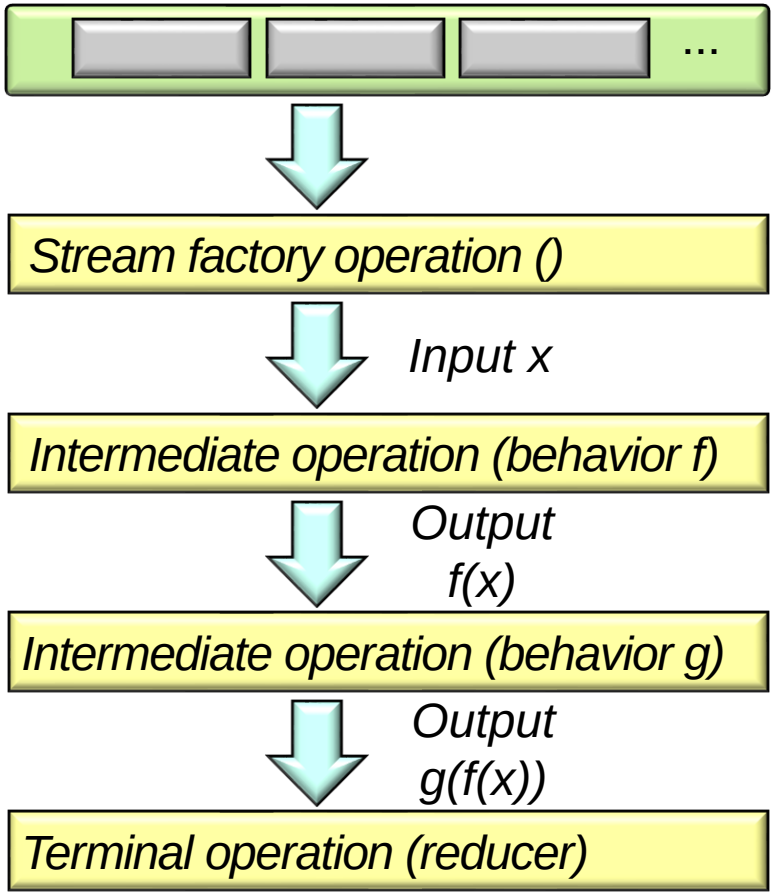
- Recall the 3 phases of a Java stream
  - *Split* – Uses a spliterator to convert a data source into a stream
  - *Apply* – Process the elements in the stream
  - *Combine* – Trigger intermediate operation processing & create a single result



Why Knowledge of Streams Internals Matters

- Recall the 3 phases of a Java stream
  - *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



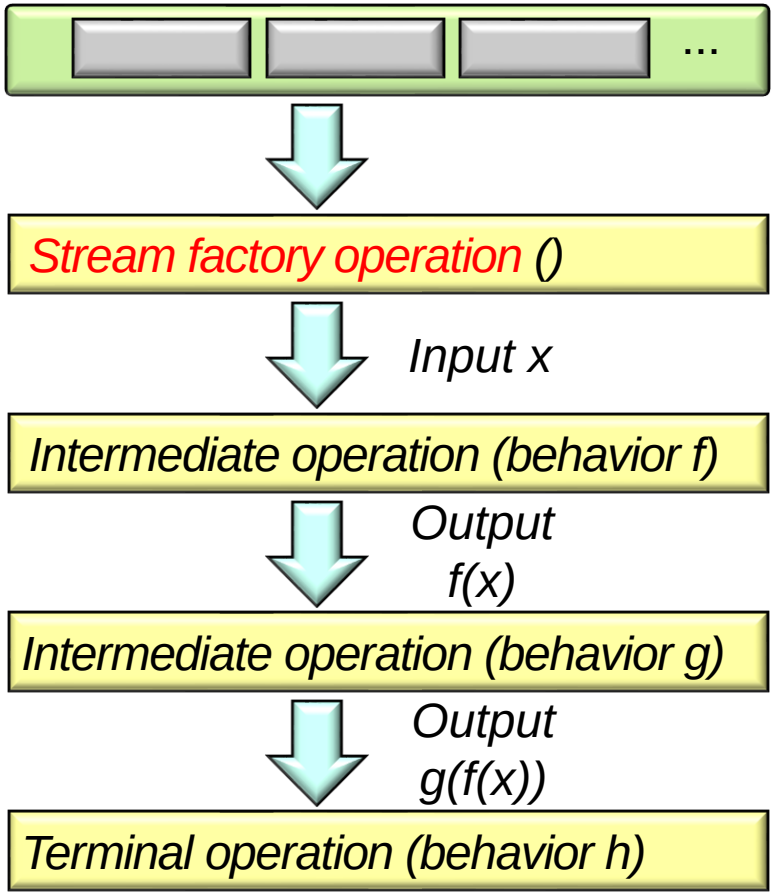


Knowing which of these phases you can control (& how) is important!

Java Streams Splitting & Combining Mechanisms

Java Streams Splitting & Combining Mechanisms

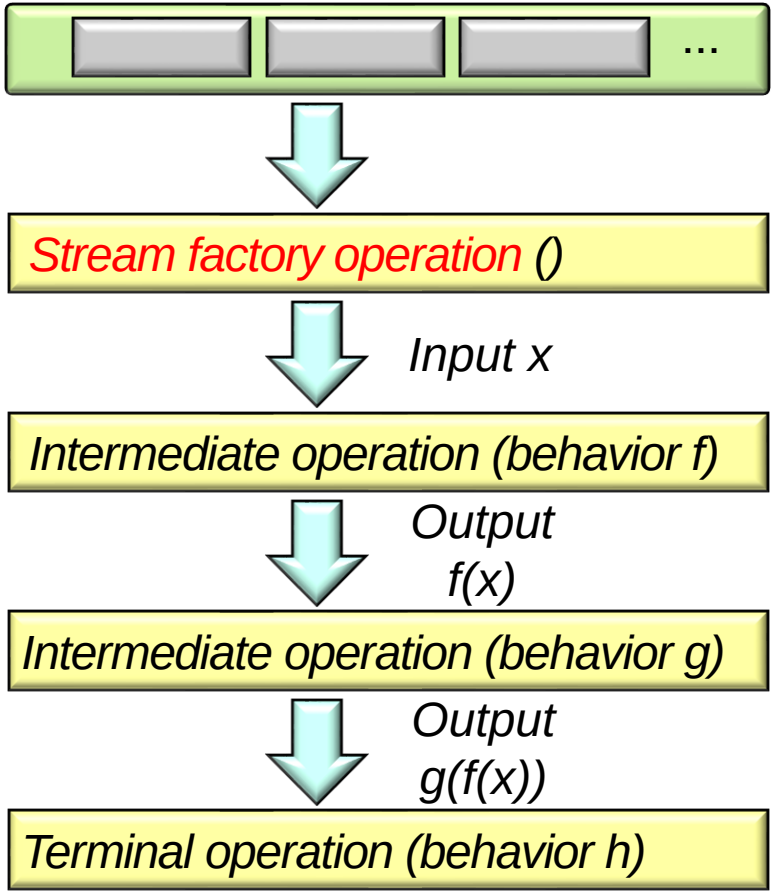
- A stream's splitting & combining mechanisms are often invisible



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);
  }
 ...
}
```

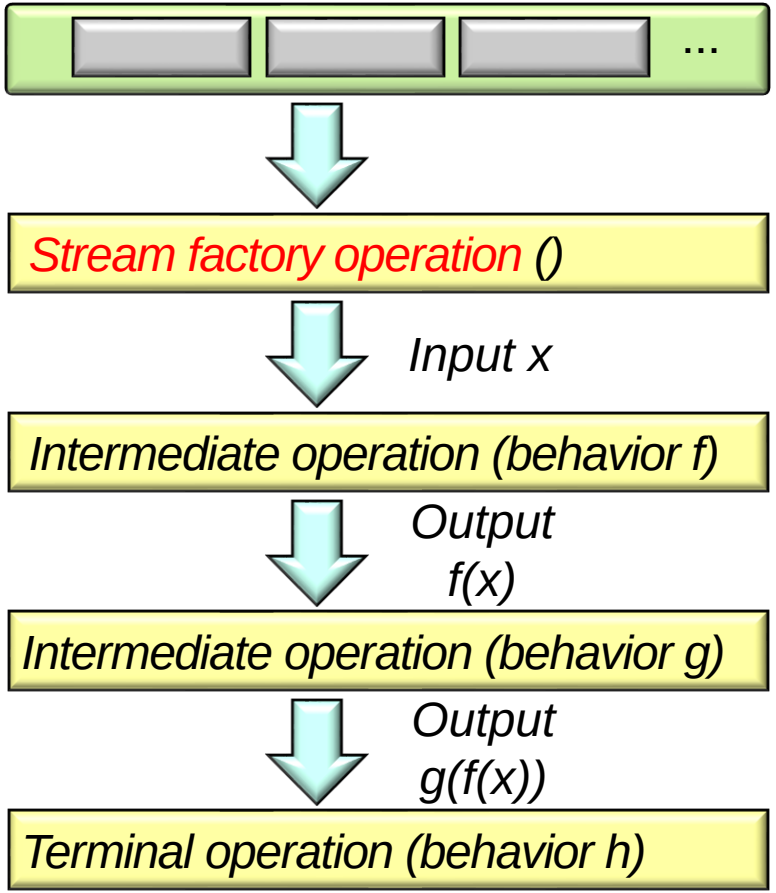


See docs.oracle.com/javase/8/docs/api/java/util/Collection.html

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);
  }
 ...
}
```

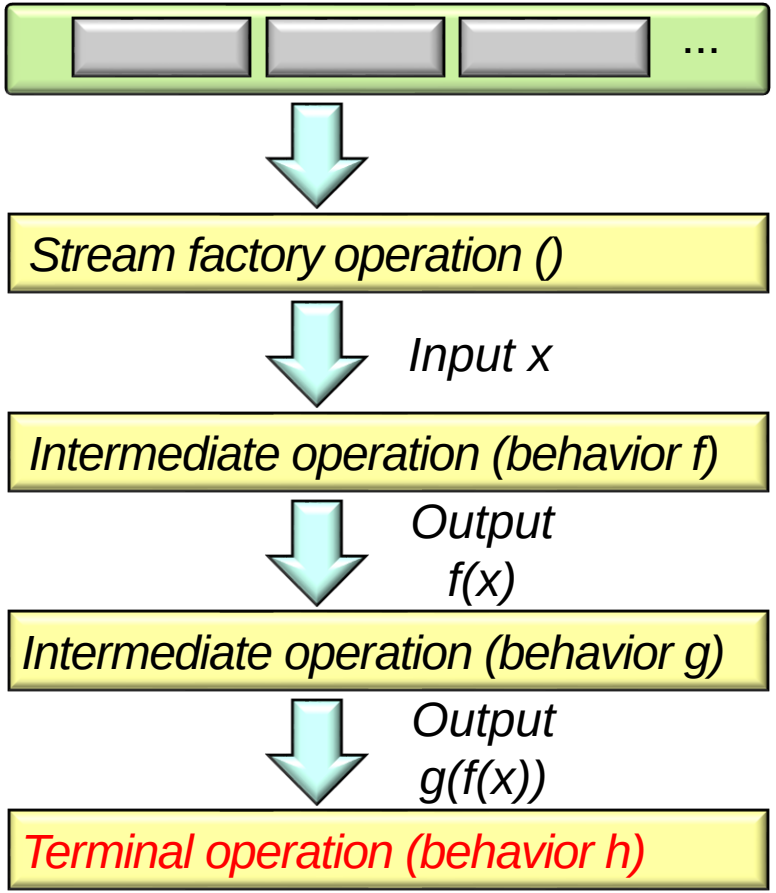


See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html

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

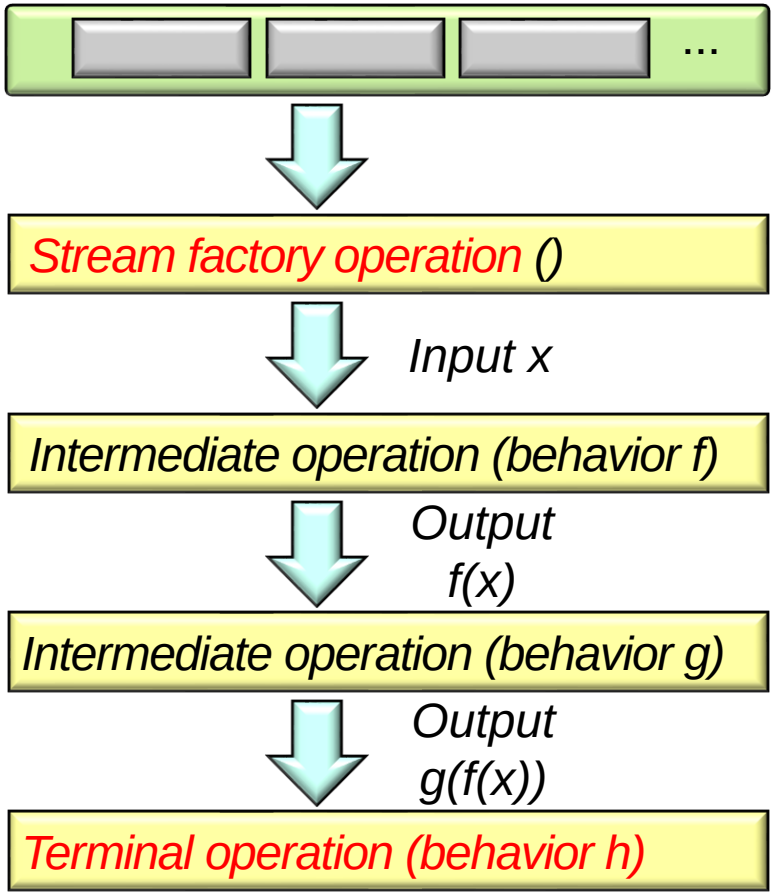
```
final class Collectors {
  ...
  public static <T> Collector<T, ?, List<T>>
    toList() { ... }

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



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

Java Streams Splitting & Combining Mechanisms

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

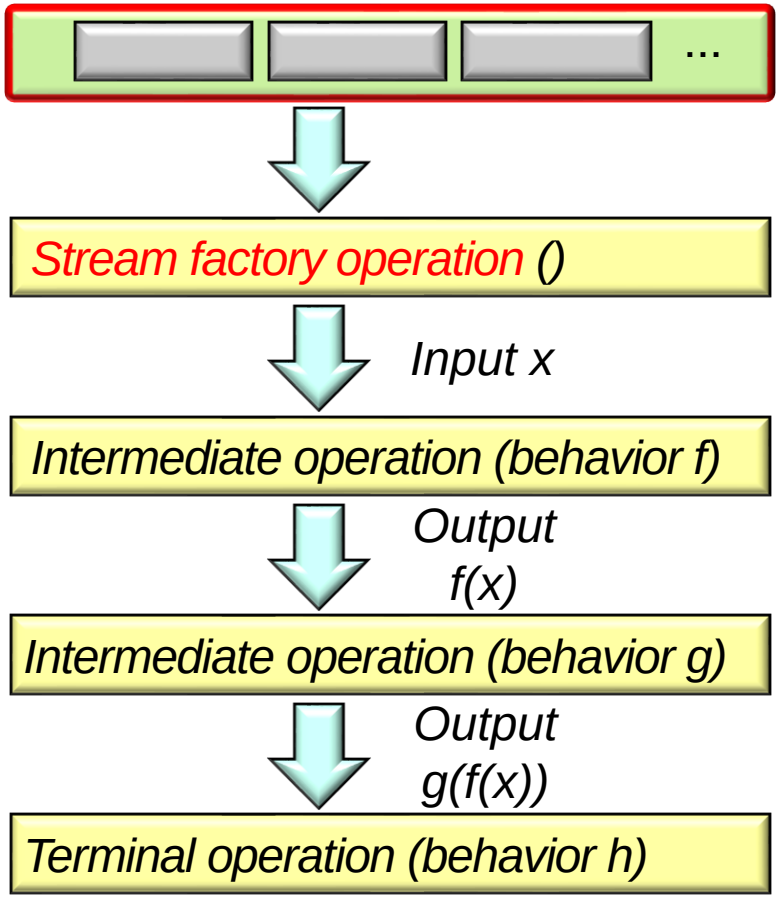


Java Streams Splitting & Combining Mechanisms

- 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.



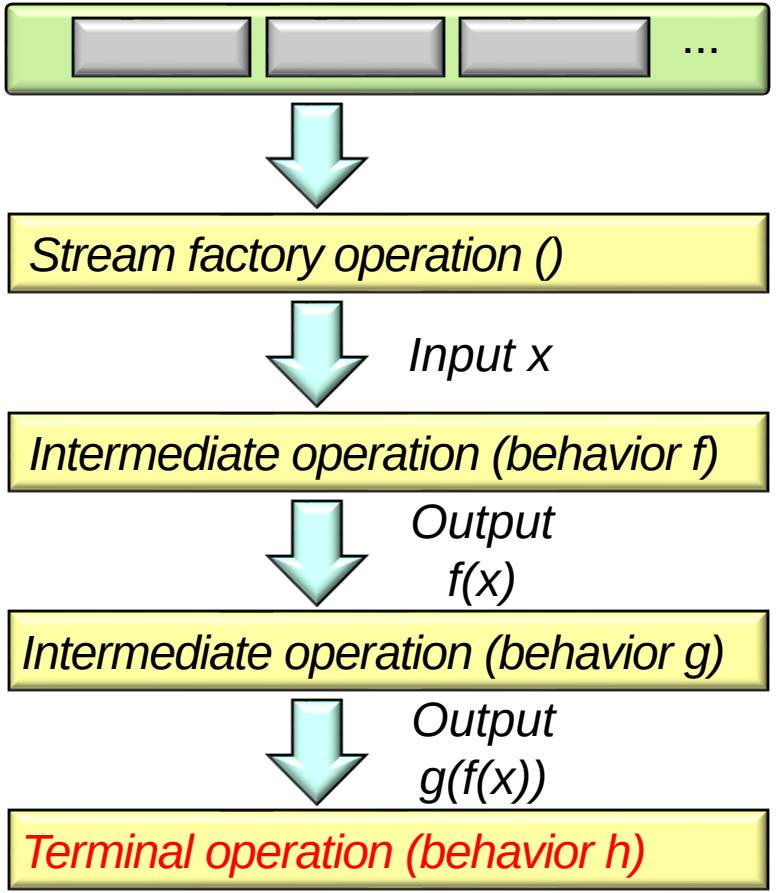
See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html

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.



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

End of Overview of Java Streams Internals (Part 1)