Java Streams: Terminal Operations

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

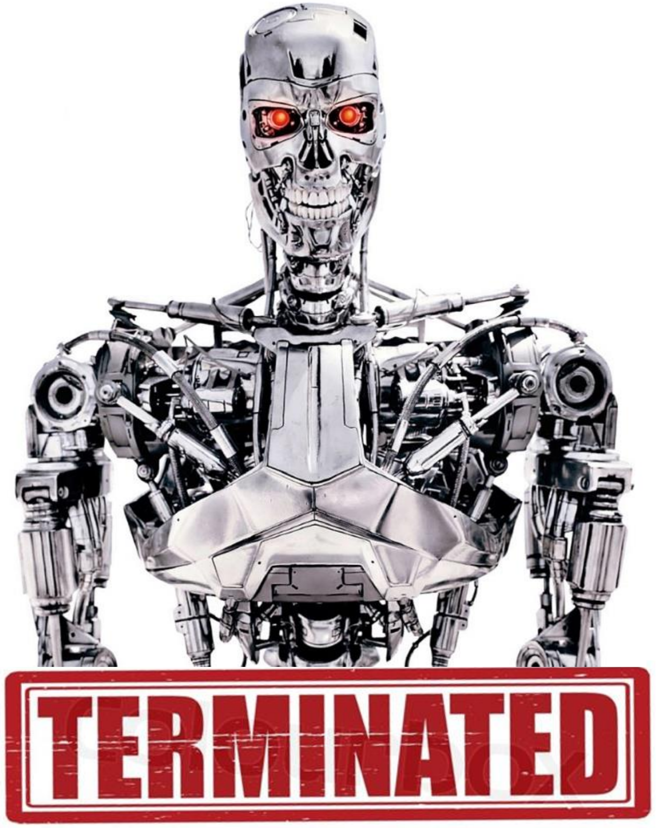
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

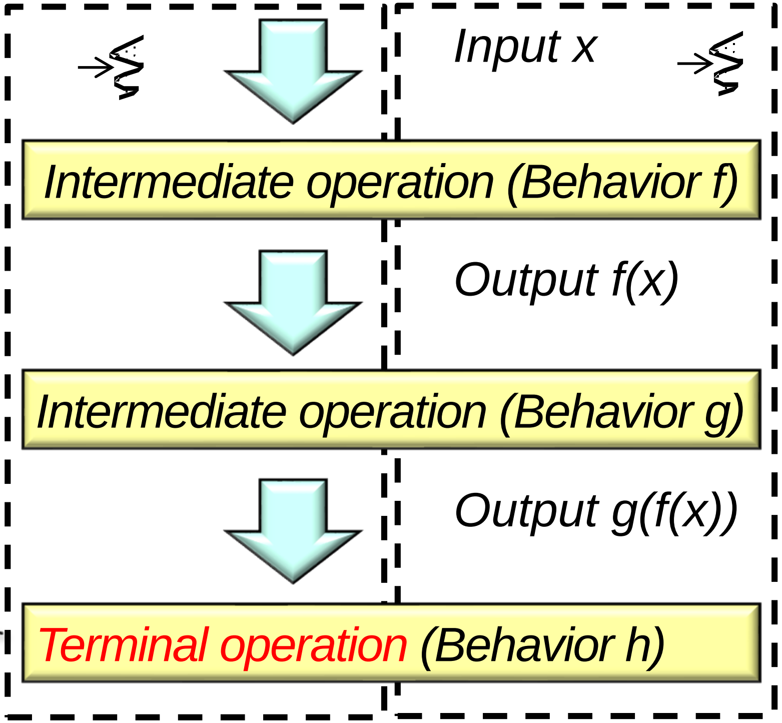
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations
  - Terminal operations



Learning Objectives in this Part of the Lesson

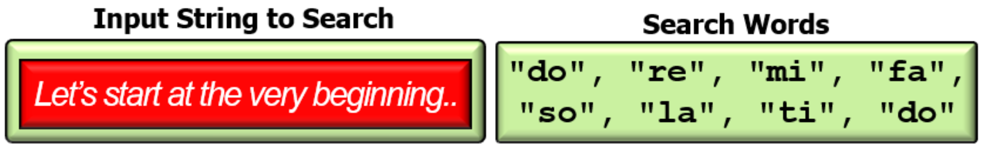
- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations
  - Terminal operations

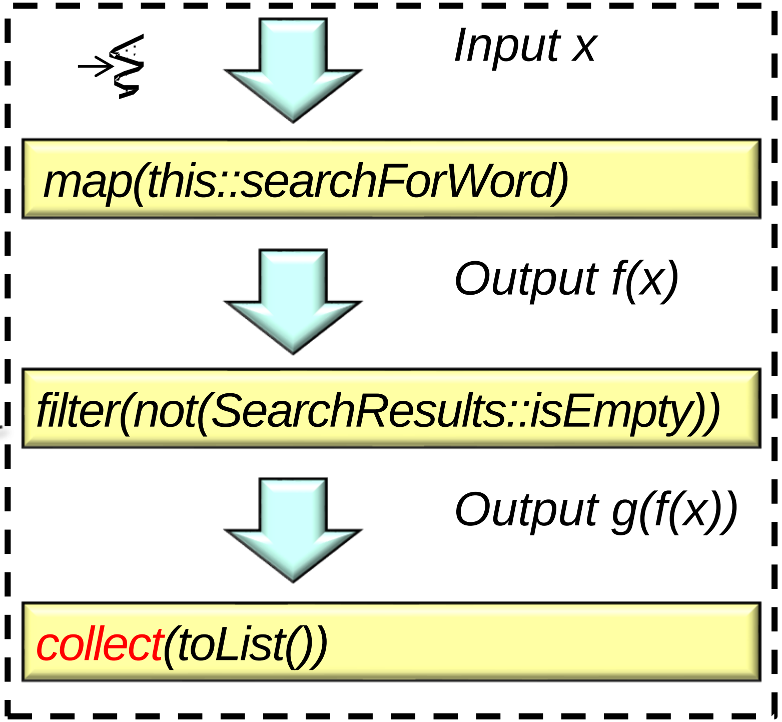

Input x
Intermediate operation (Behavior f)
Output f(x)
Intermediate operation (Behavior g)
Output g(f(x))
Terminal operation (Behavior h)

These operations also apply to both sequential & parallel streams

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations
  - Terminal operations


Input String to Search
Let's start at the very beginning..
Search Words
"do", "re", "mi", "fa", "so", "la", "ti", "do"



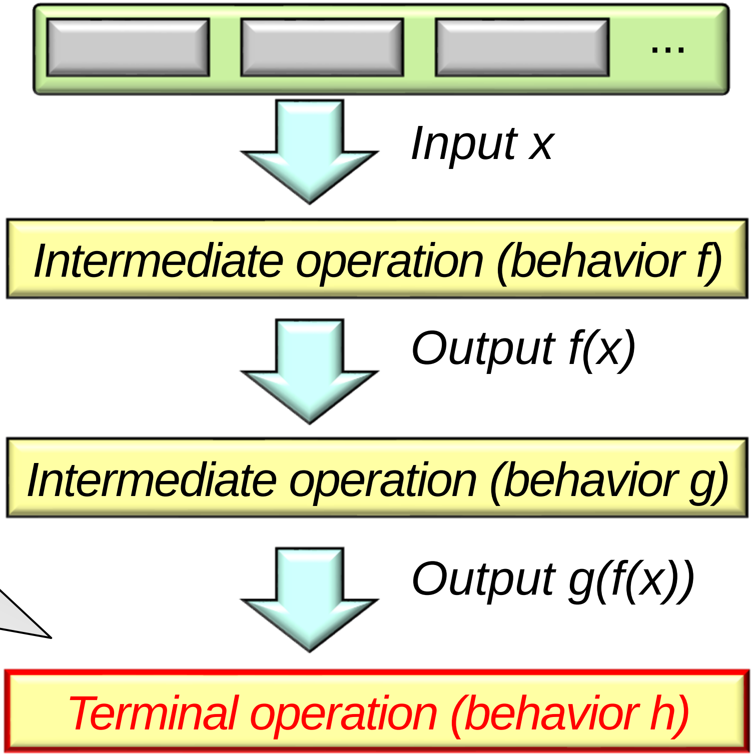
We continue to showcase the SimpleSearchStream program

Overview of Terminal Operations

Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result

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


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

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

Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  - No value at all
    - e.g., forEach() & forEachOrdered()

forEach() only has side-effects!

Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  - No value at all
    - e.g., forEach() & forEachOrdered()

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

Print each character in Hamlet that starts with 'H' or 'h' in consistently capitalized & sorted order.

Overview of Common Stream Terminal Operations

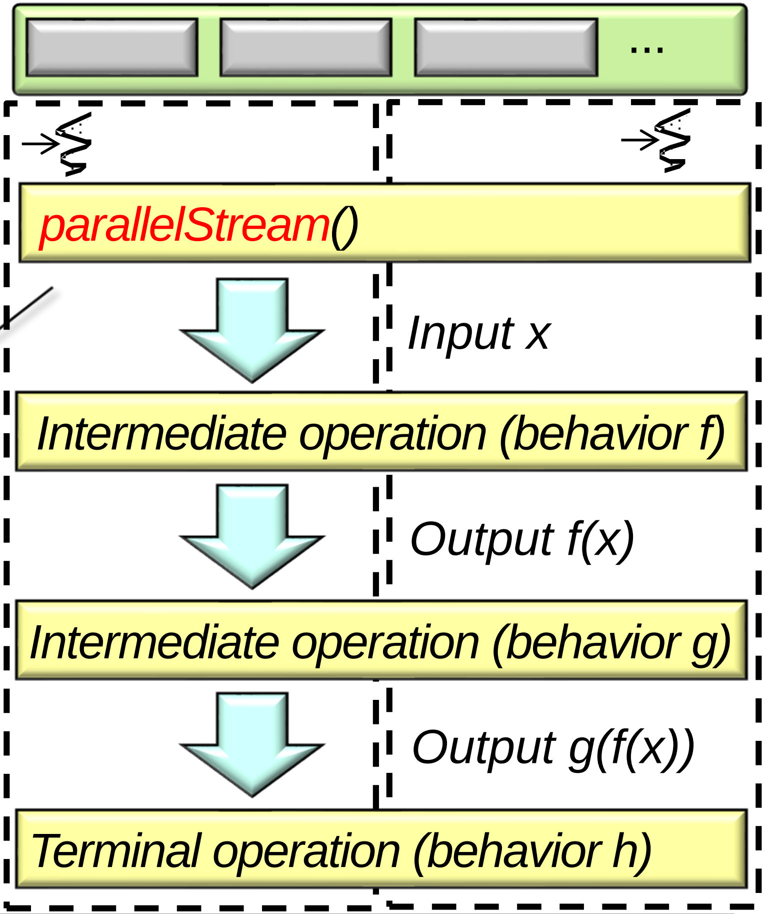
- Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  - No value at all
  - The result of a reduction operation
    - e.g., collect() & reduce()

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

Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  - No value at all
  - The result of a reduction operation
    - e.g., collect() & reduce()

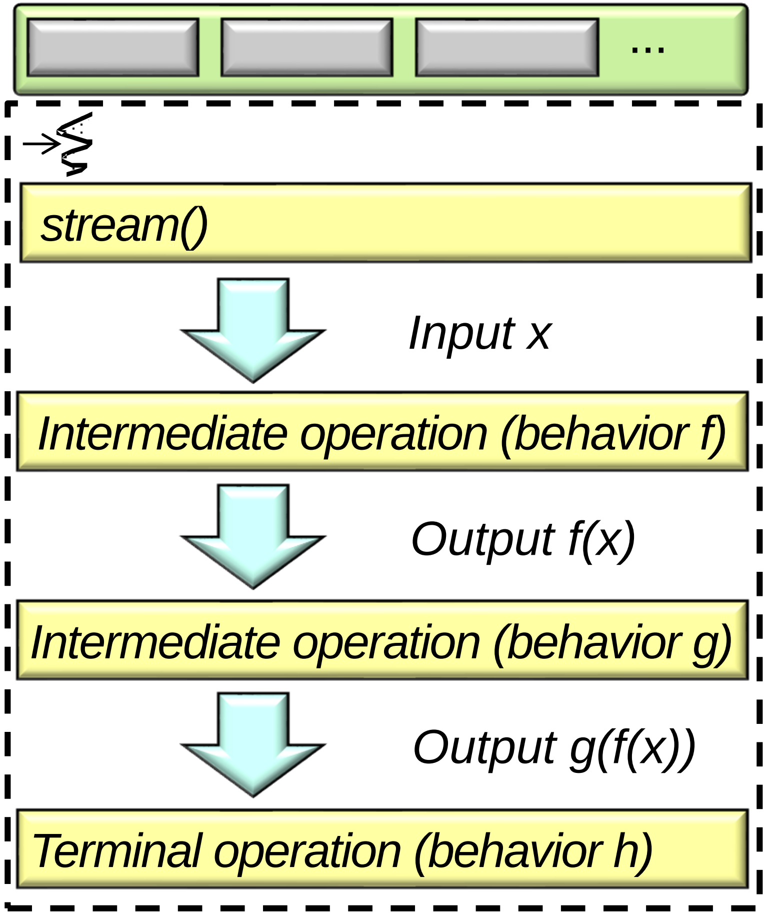
collect() & reduce() terminal operations work seamlessly with parallel streams.


...
parallelStream()
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/tutorial/collections/streams/parallelism.html

Overview of the collect() Terminal Operation

- A terminal operation also triggers all the intermediate operation processing

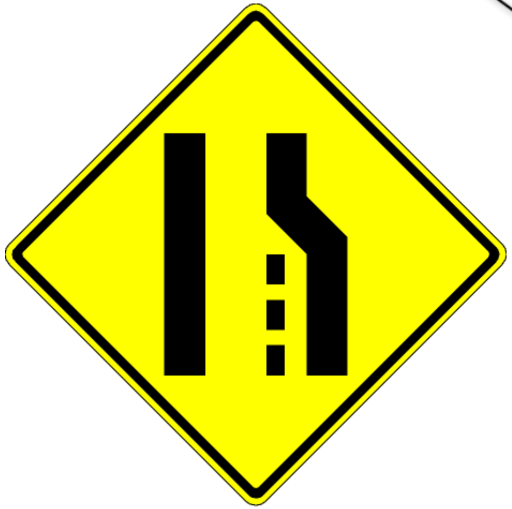

...
stream()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (behavior h)

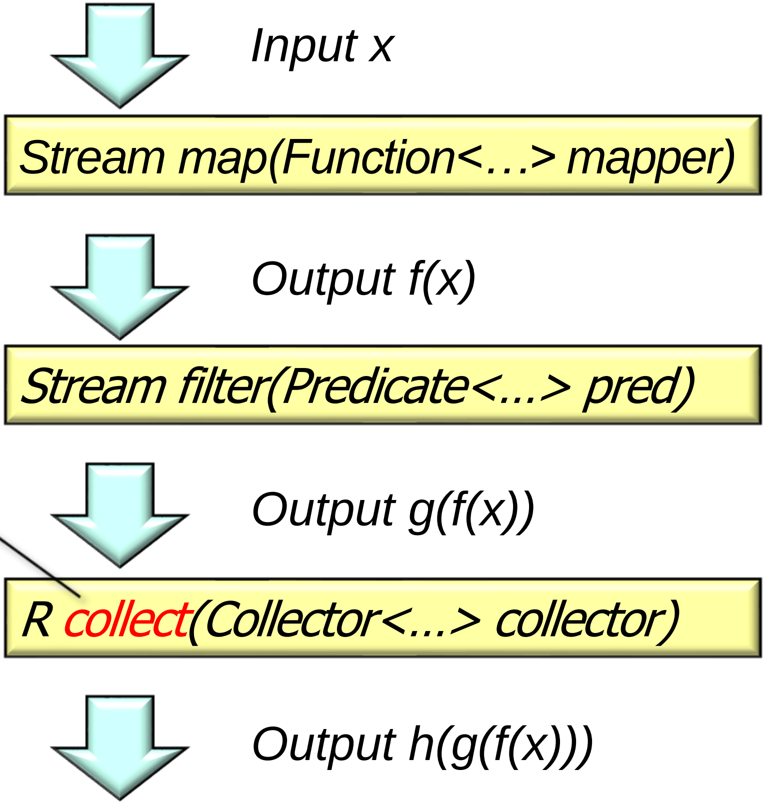
Overview of the collect() Terminal Operation

Overview of the collect() Terminal Operation

- Overview of the collect() terminal operation

This terminal operation uses a collector to perform a reduction on the elements of its input stream & returns the results of the reduction.

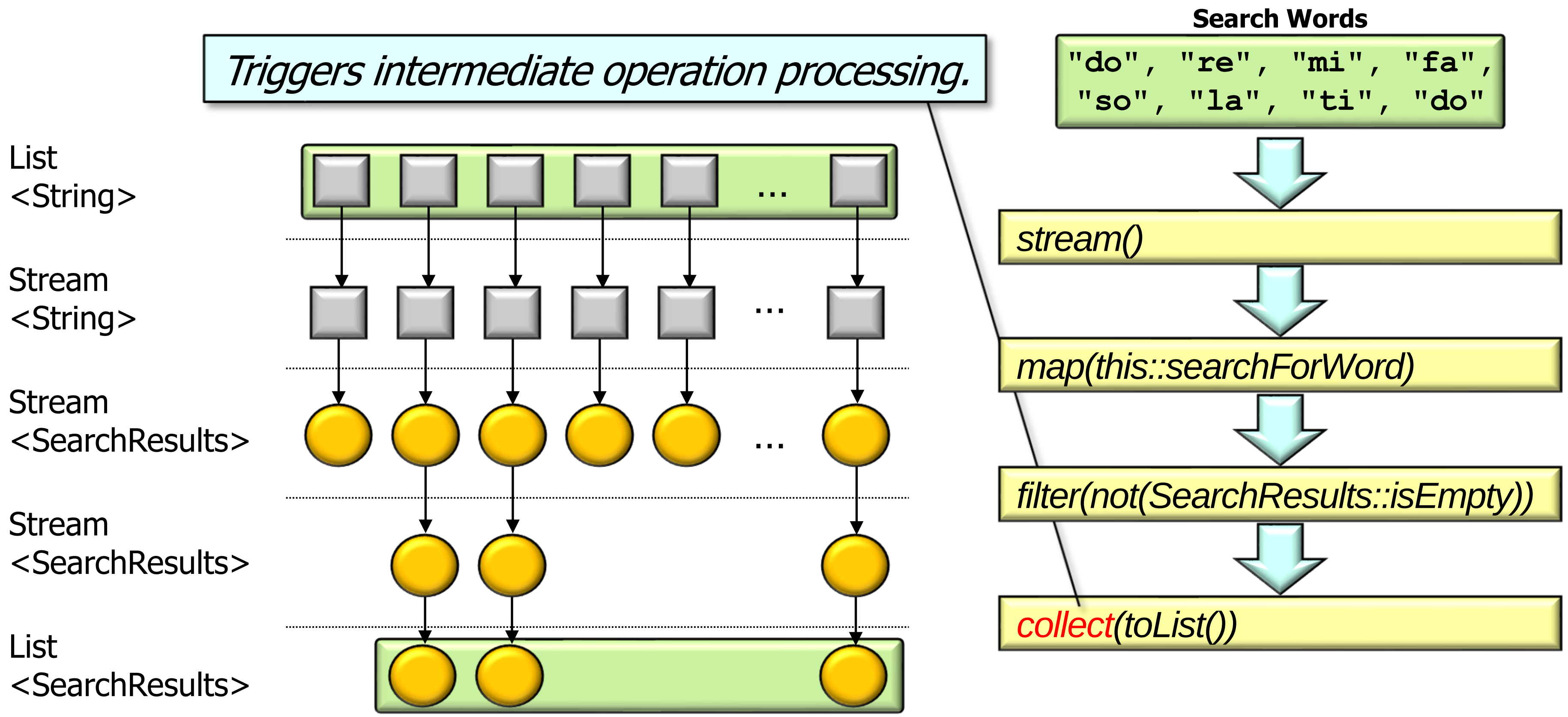



Input x
Stream map(Function<...> mapper)
Output f(x)
Stream filter(Predicate<...> pred)
Output g(f(x))
R collect(Collector<...> collector)
Output h(g(f(x)))

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

Overview of the collect() Terminal Operation

- Overview of the collect() terminal operation

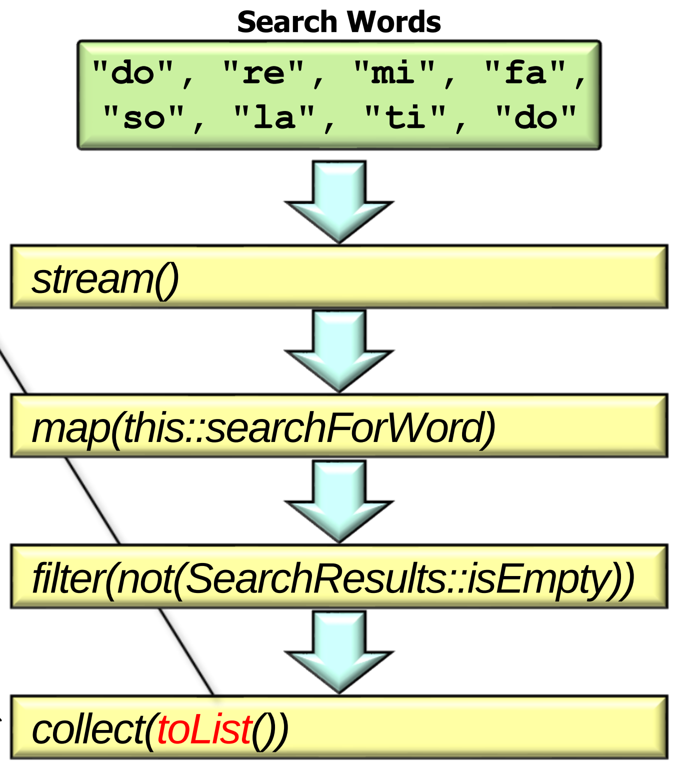


Overview of the collect() Terminal Operation

- Overview of the collect() terminal operation

Create a list of non-empty SearchResults.

```
List<SearchResults> results =
  wordsToFind
    .stream()
    .map(this::searchForWord)
    .filter(not
       (SearchResults::isEmpty))
    .collect(toList());
```



The list returned from collect() presents search results in "encounter order"

End of Java Streams: Terminal Operations