Overview of Java 8 Streams (Part 2)

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

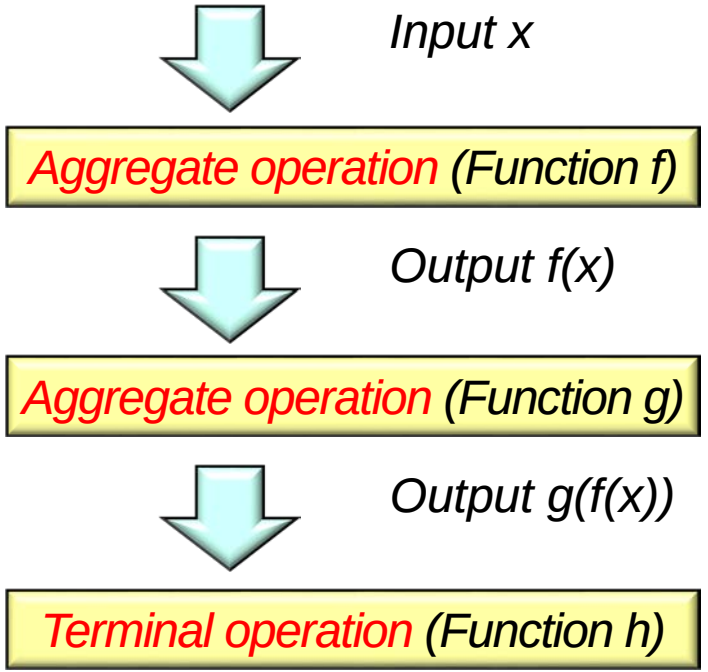
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

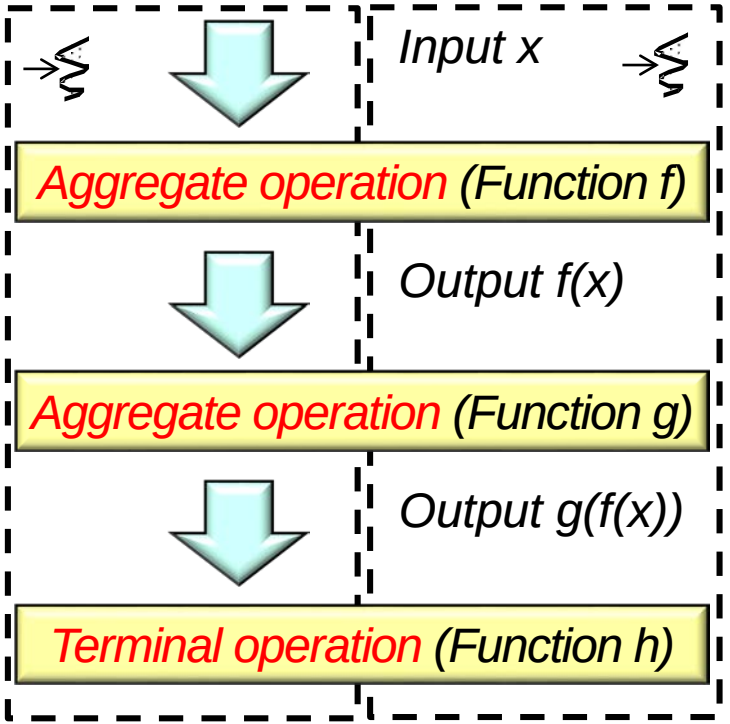
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations



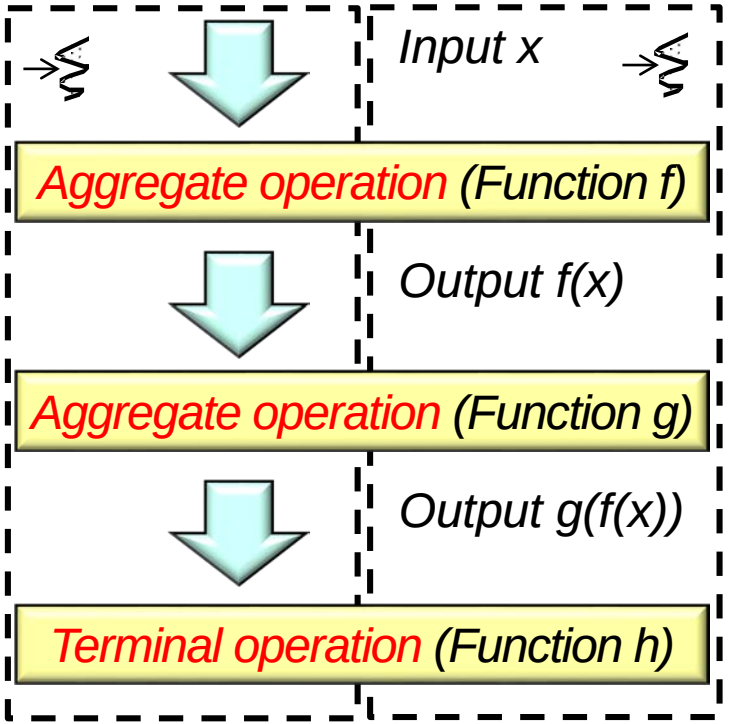
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations
    - These operations apply to both sequential & parallel streams



Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations
    - These operations apply to both sequential & parallel streams

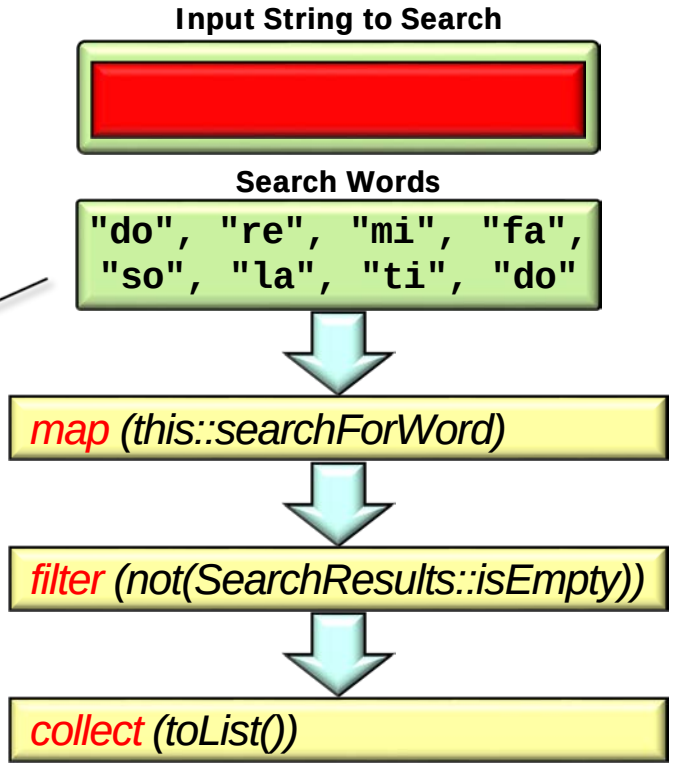


Being a good streams programmer makes you a better parallel streams programmer

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations
    - These operations apply to both sequential & parallel streams

We'll use a simple sequential stream example to explain common Java 8 aggregate operations

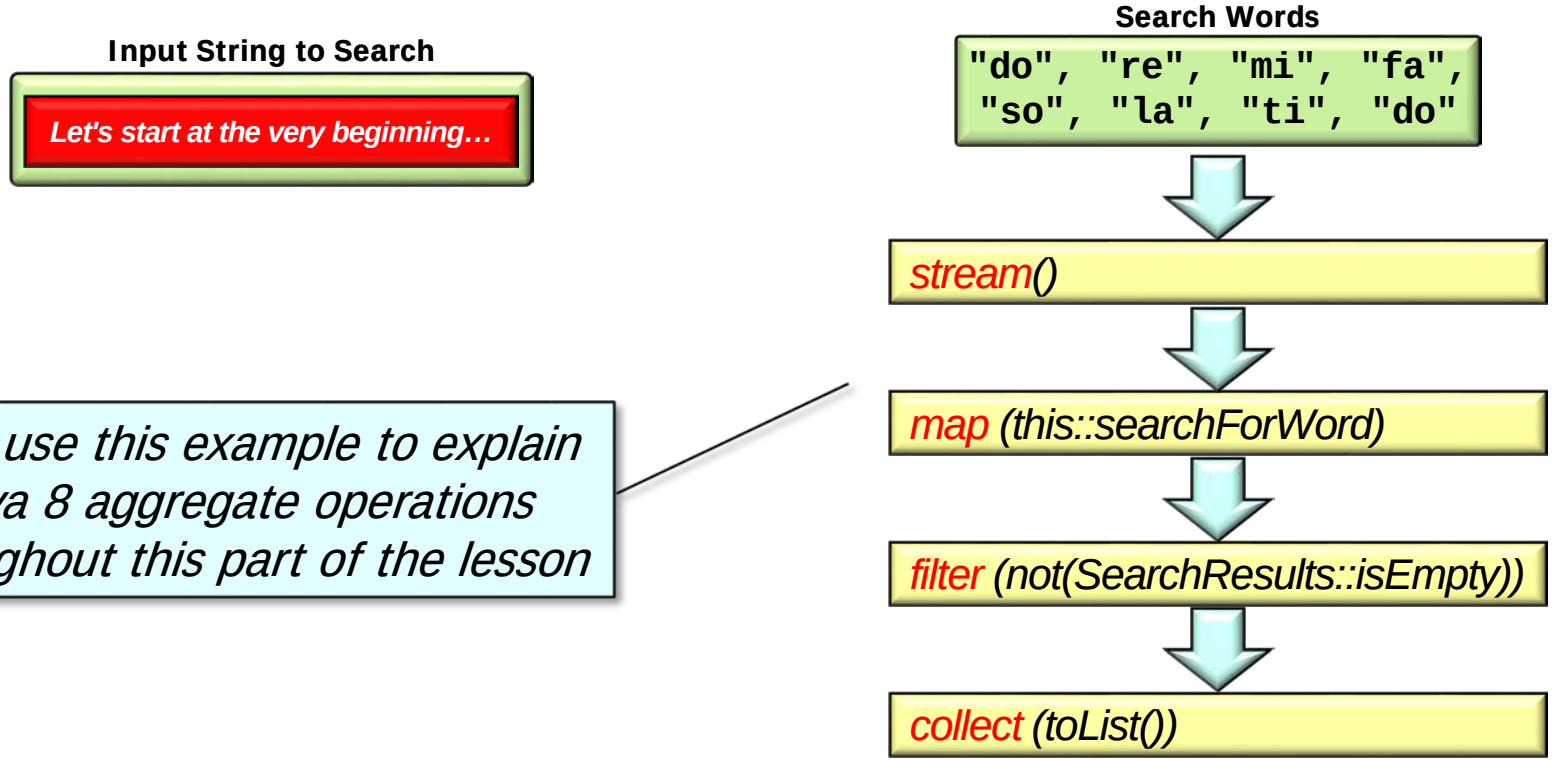


See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream

Overview of SimpleSearch Stream Example

Overview of SimpleSearchStream Example

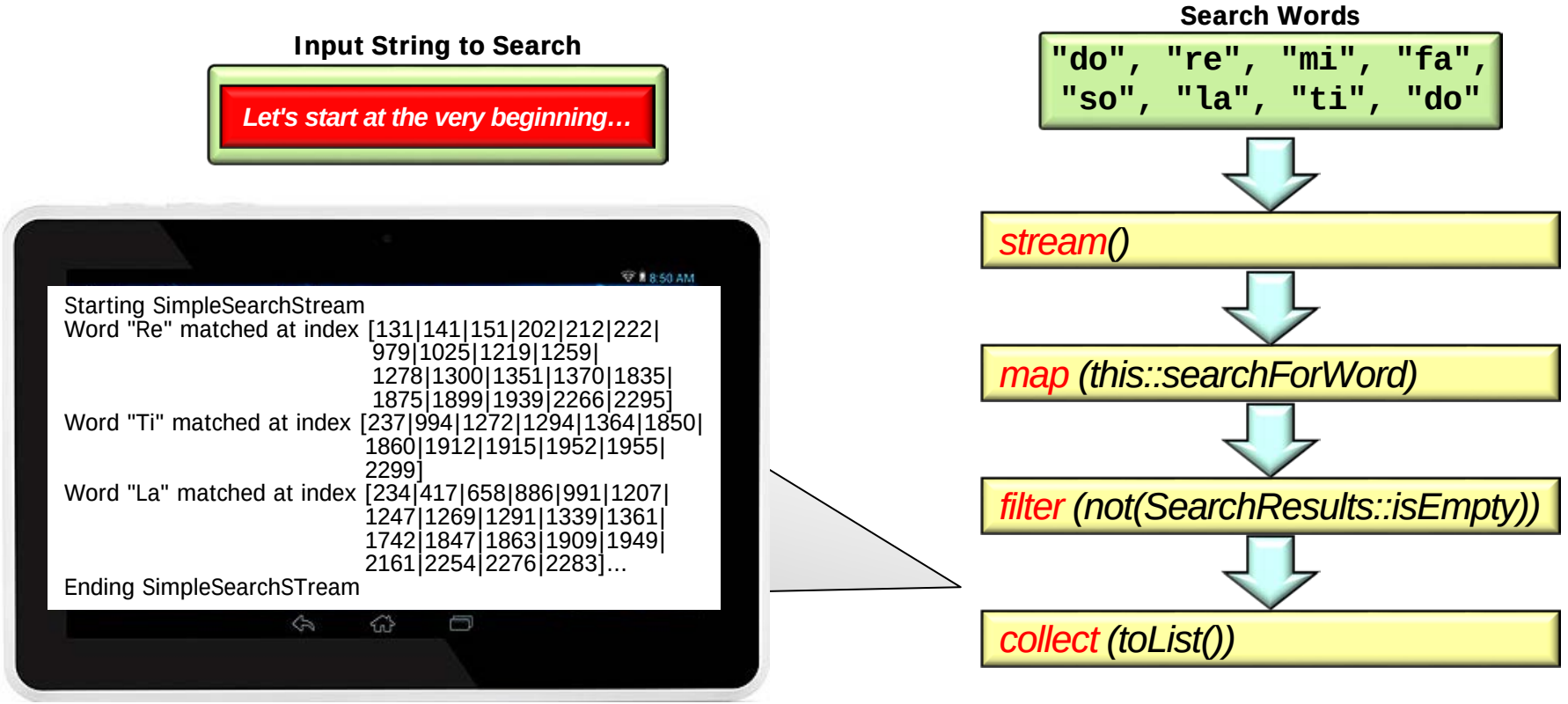
- This example finds words in an input string



See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream

Overview of SimpleSearchStream Example

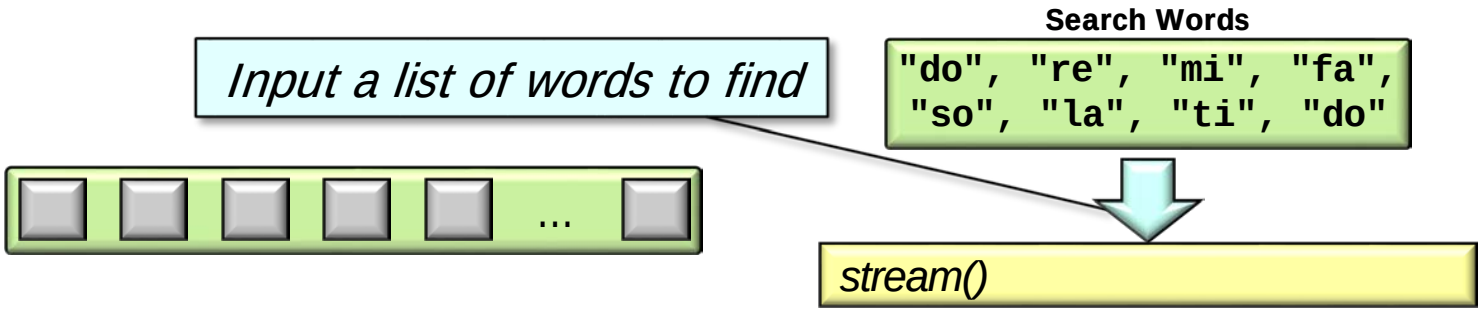
- This example finds words in an input string



See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream

Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string

List
<String>

Search Words

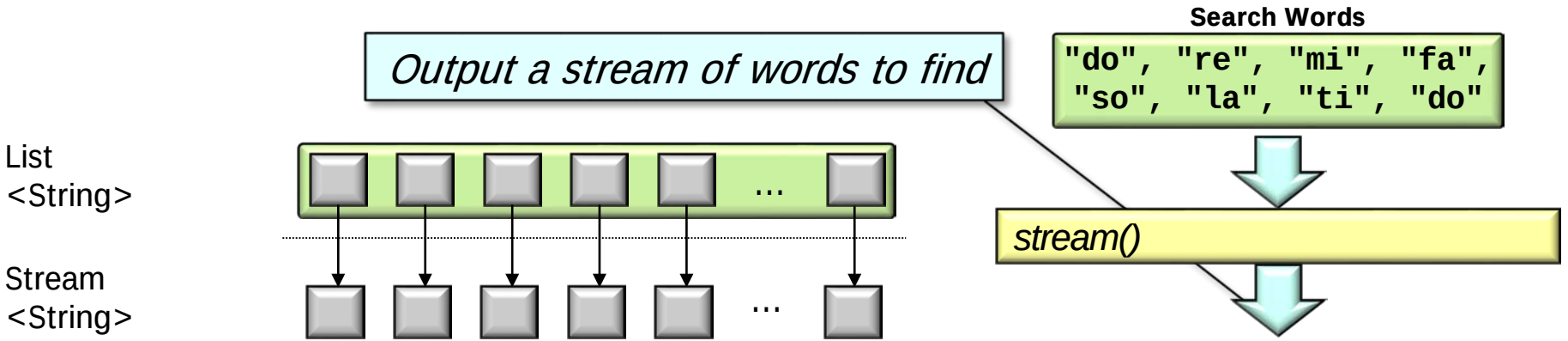
```
"do", "re", "mi", "fa",
"so", "la", "ti", "do"
```

stream()

Convert collection to a (sequential) stream

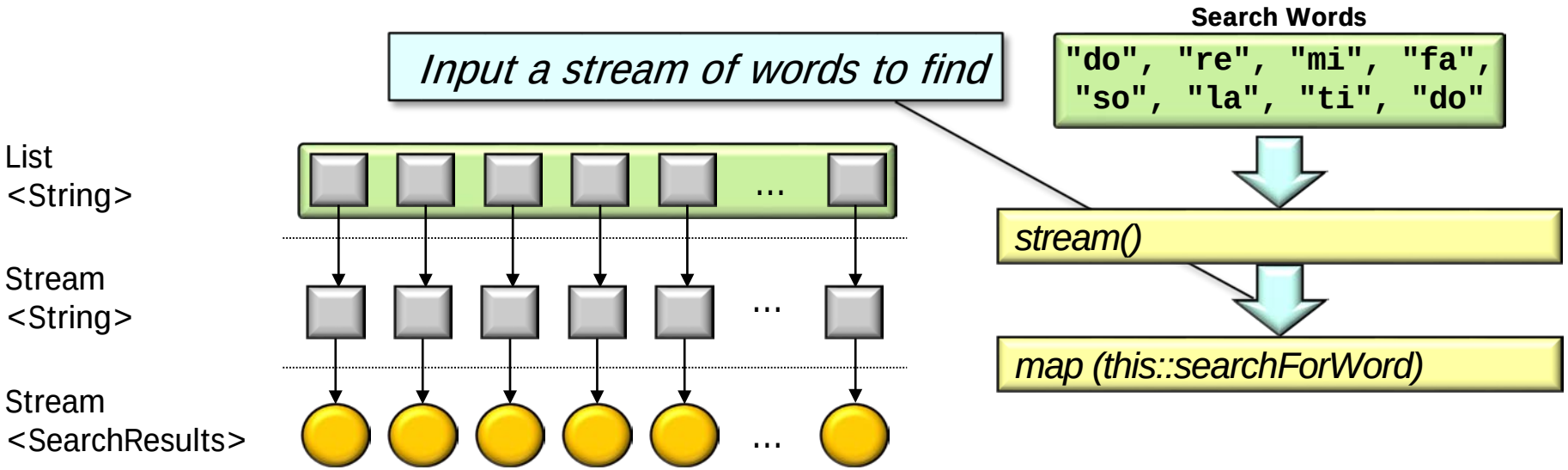
Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string



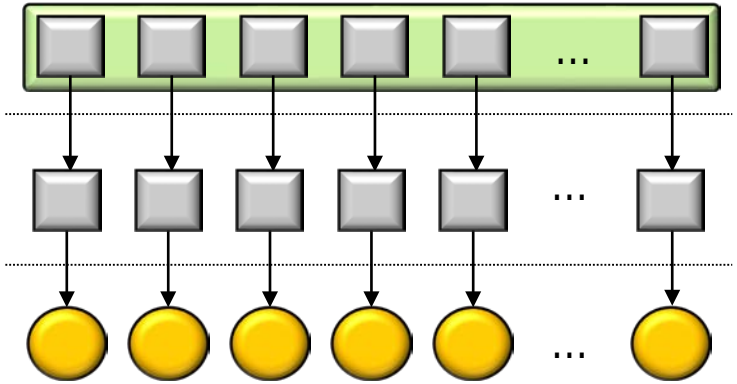
Overview of SimpleSearchStream Example

- This example finds words in an input string

List
<String>

Stream
<String>

Stream
<SearchResults>



Search Words

```
"do", "re", "mi", "fa",
"so", "la", "ti", "do"
```

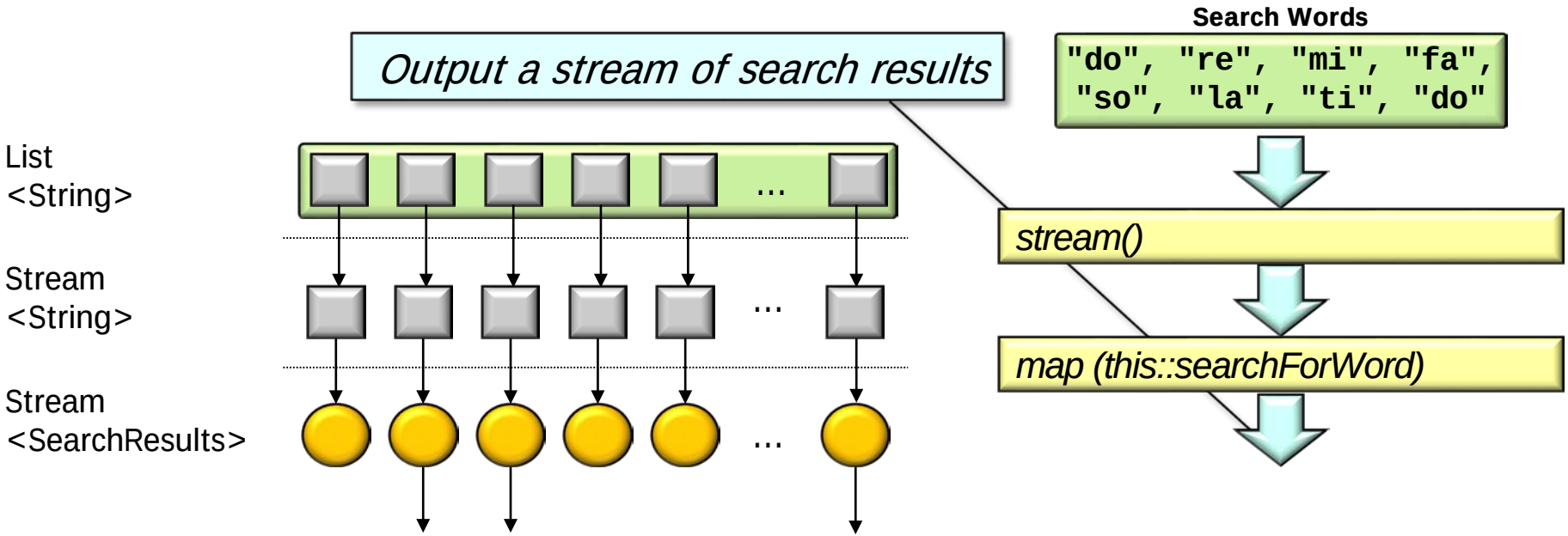
stream()

map (this::searchForWord)

Search for the word in each input string

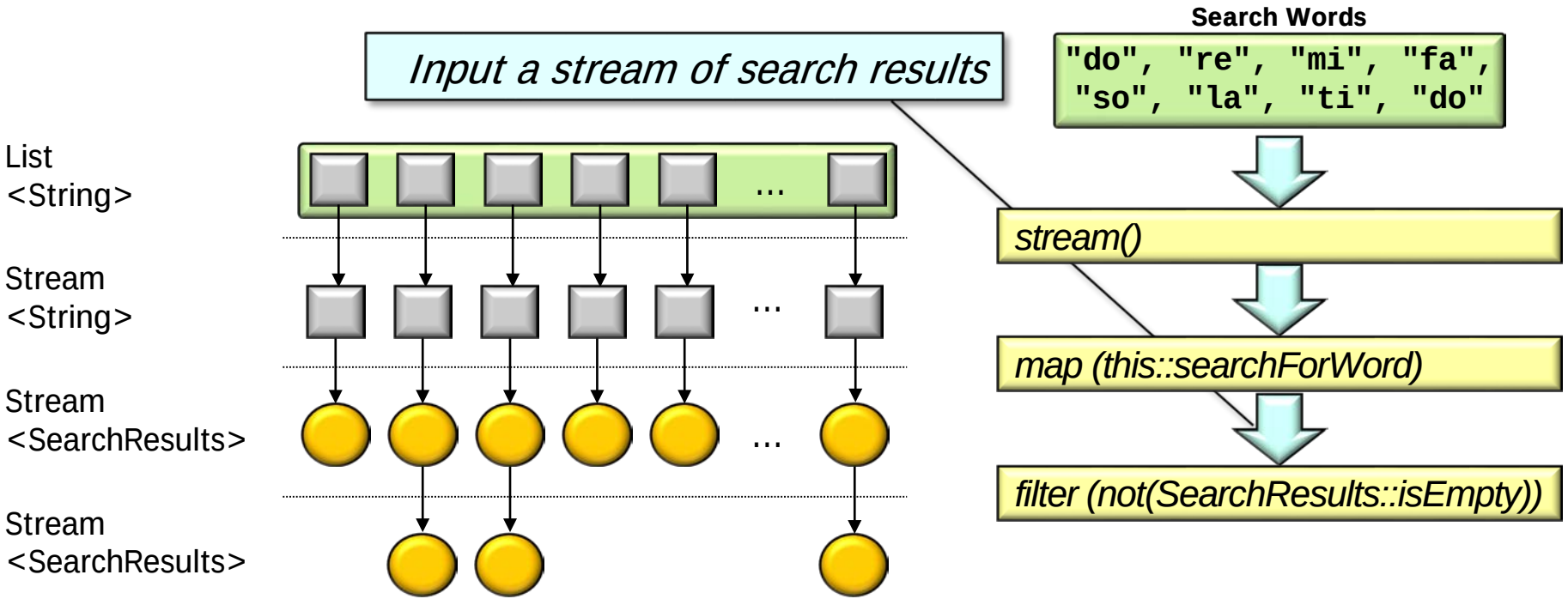
Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

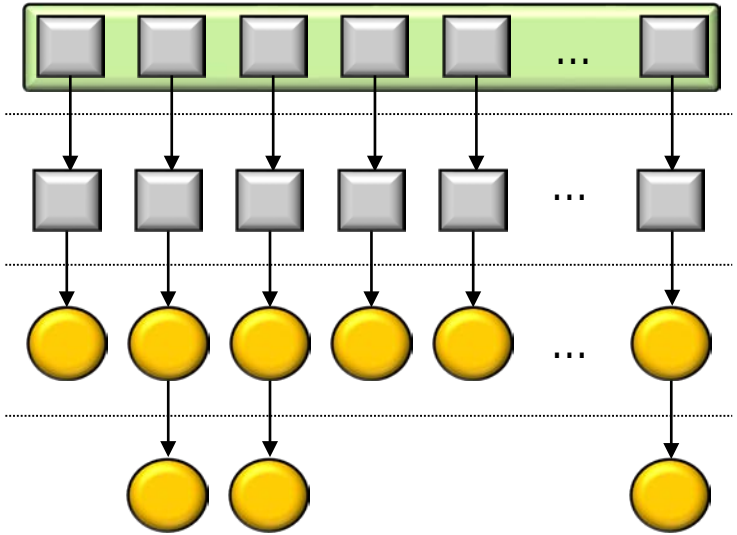
- This example finds words in an input string

List
<String>

Stream
<String>

Stream
<SearchResults>

Stream
<SearchResults>



Search Words

```
"do", "re", "mi", "fa",
"so", "la", "ti", "do"
```

stream()

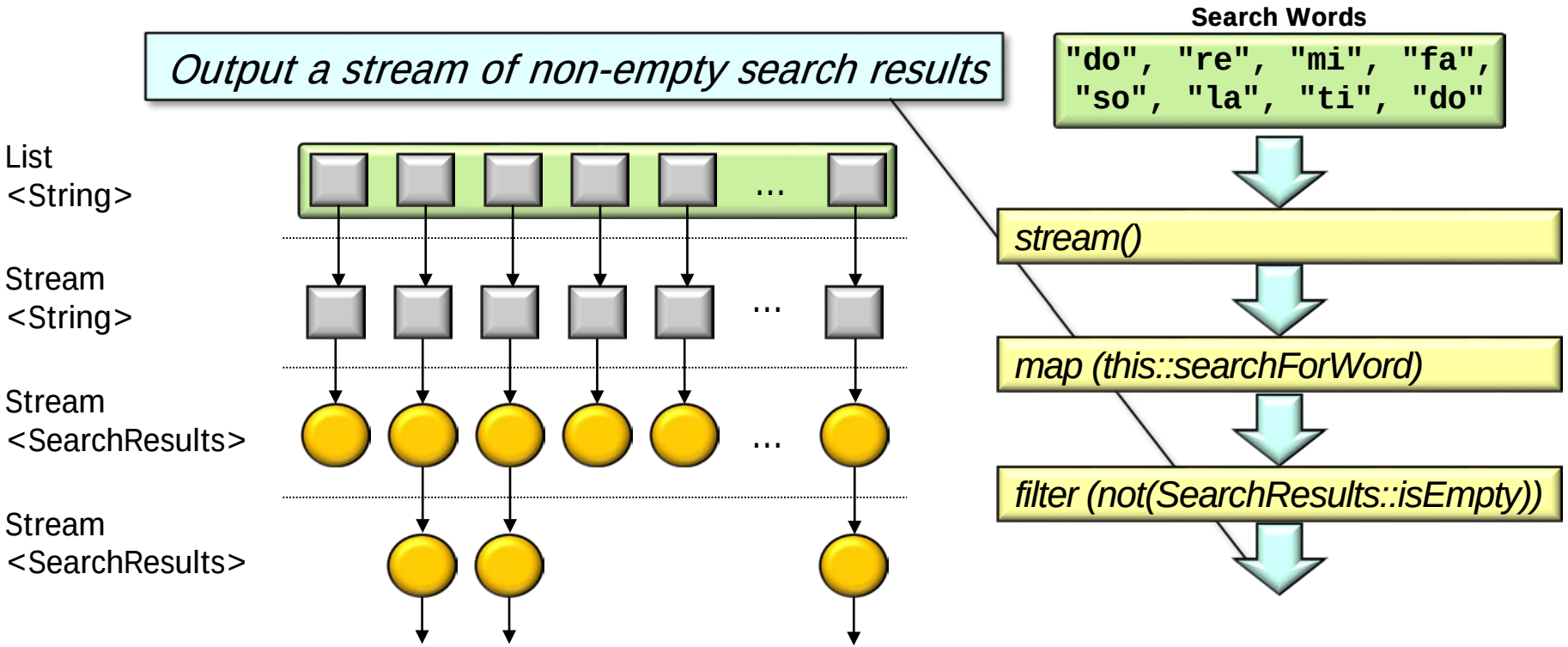
map (this::searchForWord)

filter (not(SearchResults::isEmpty))

Remove empty search results from the stream

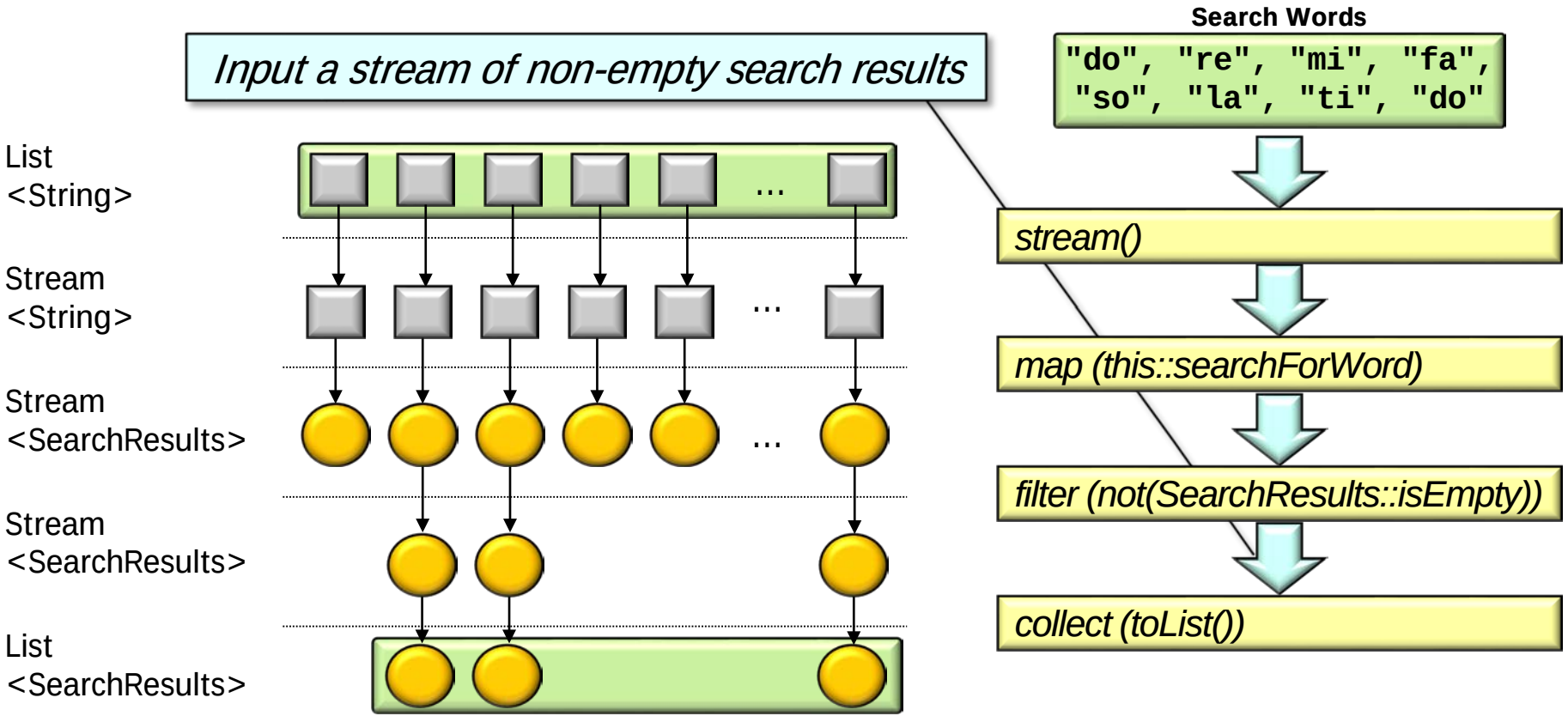
Overview of SimpleSearchStream Example

- This example finds words in an input string



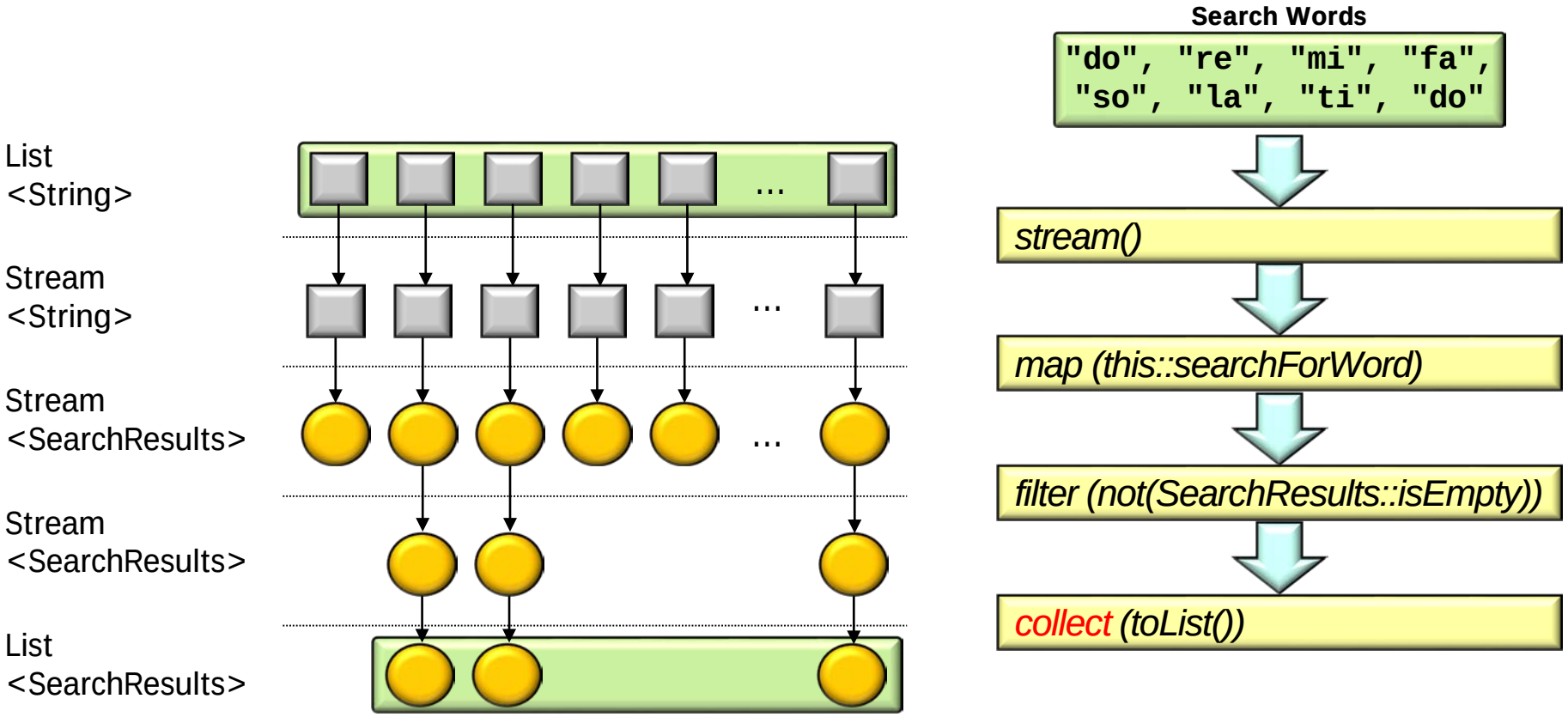
Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string



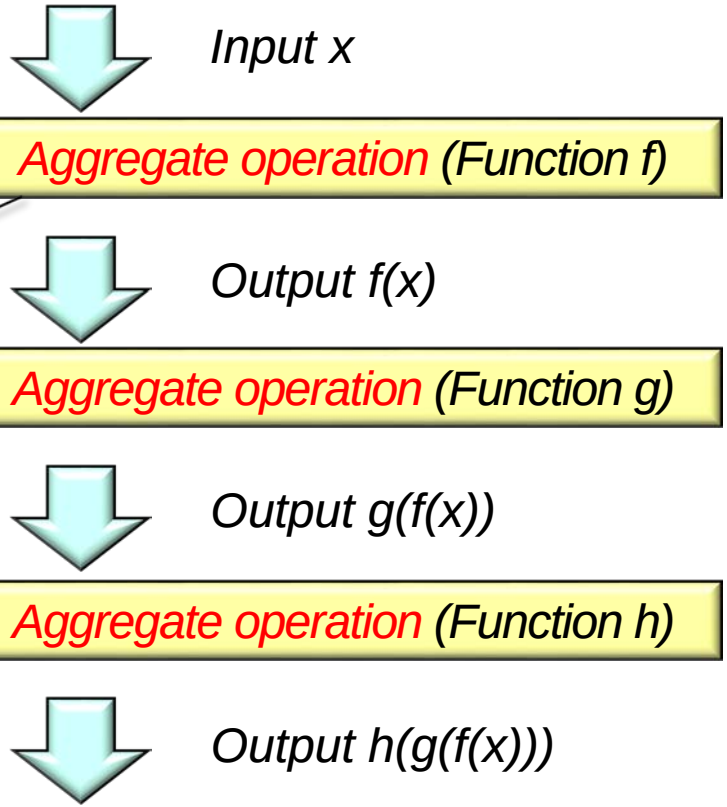
Trigger intermediate operation processing & return a list of SearchResults

Overview of Common Stream Aggregate Operations

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

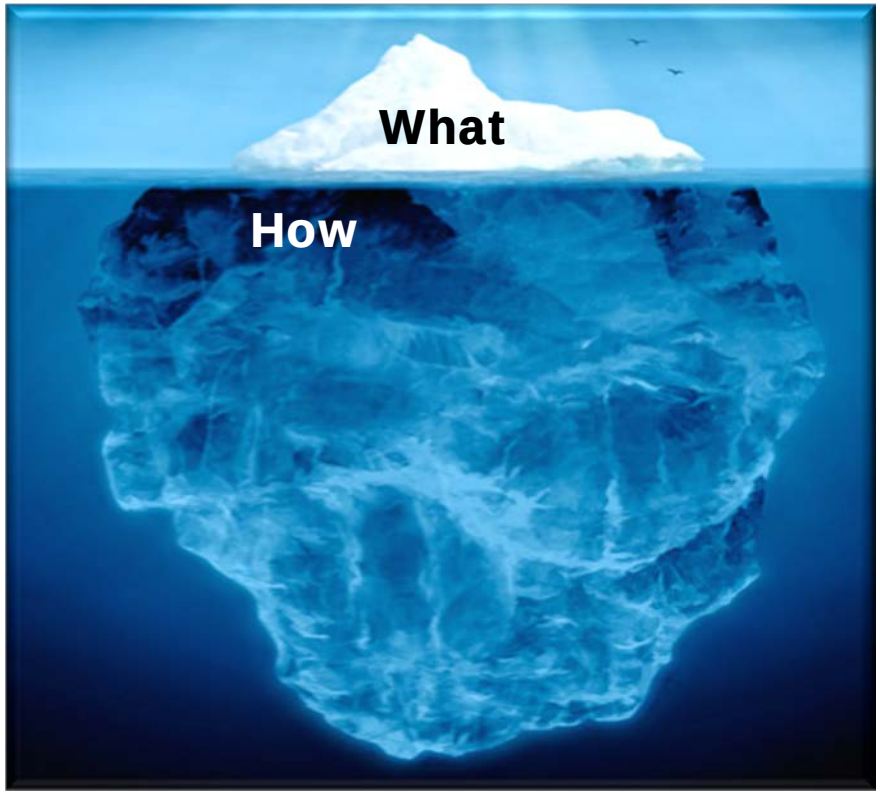
An aggregate operation is a higher-order function that applies a "behavior" (e.g., function or predicate) parameter to every element in a stream

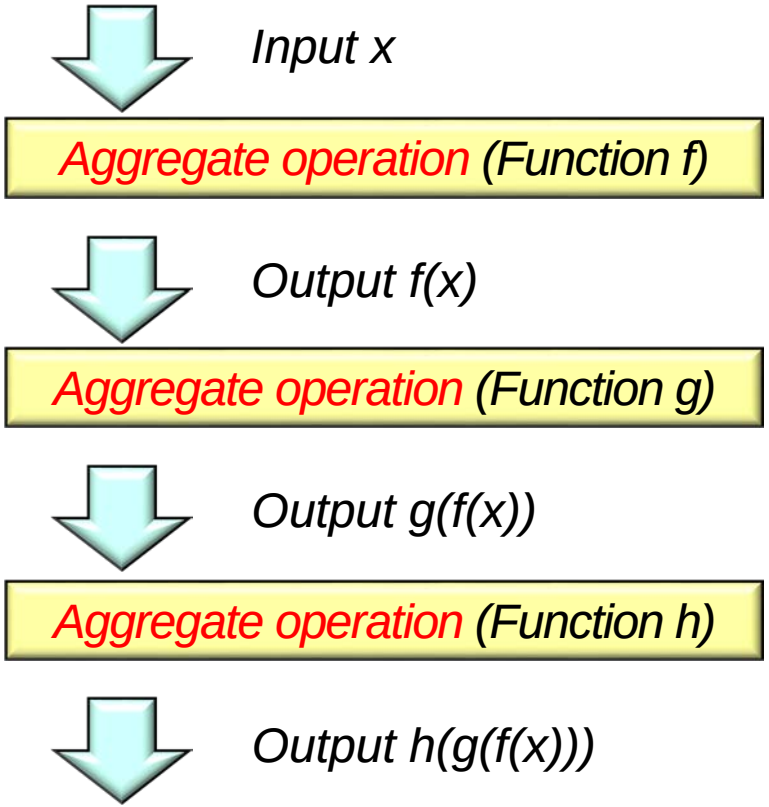


See en.wikipedia.org/wiki/Higher-order_function

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

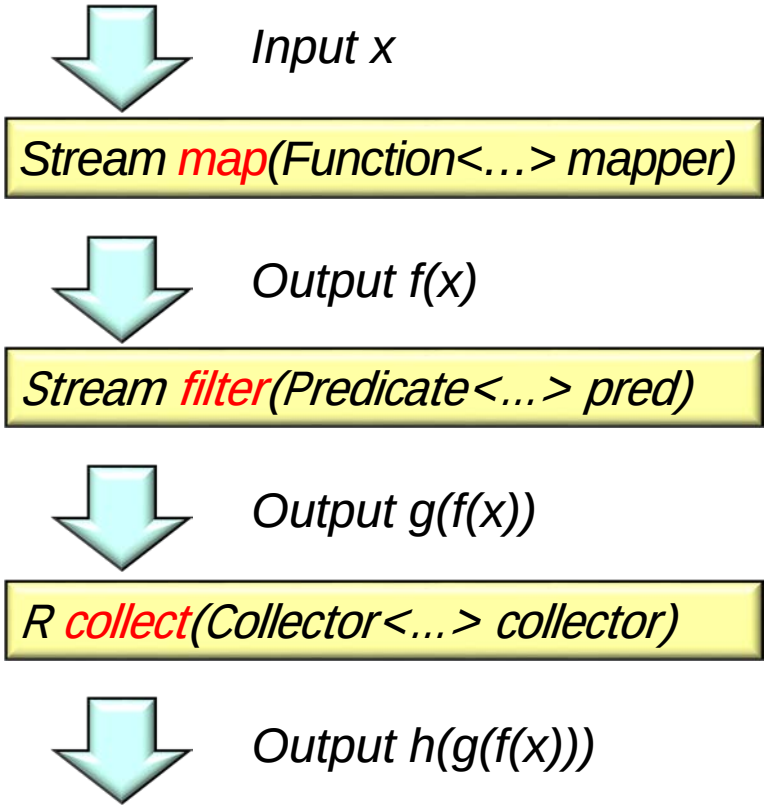




The focus is on the “what” (declarative), *not* the “how” (imperative)

Overview of Common Stream Aggregate Operations

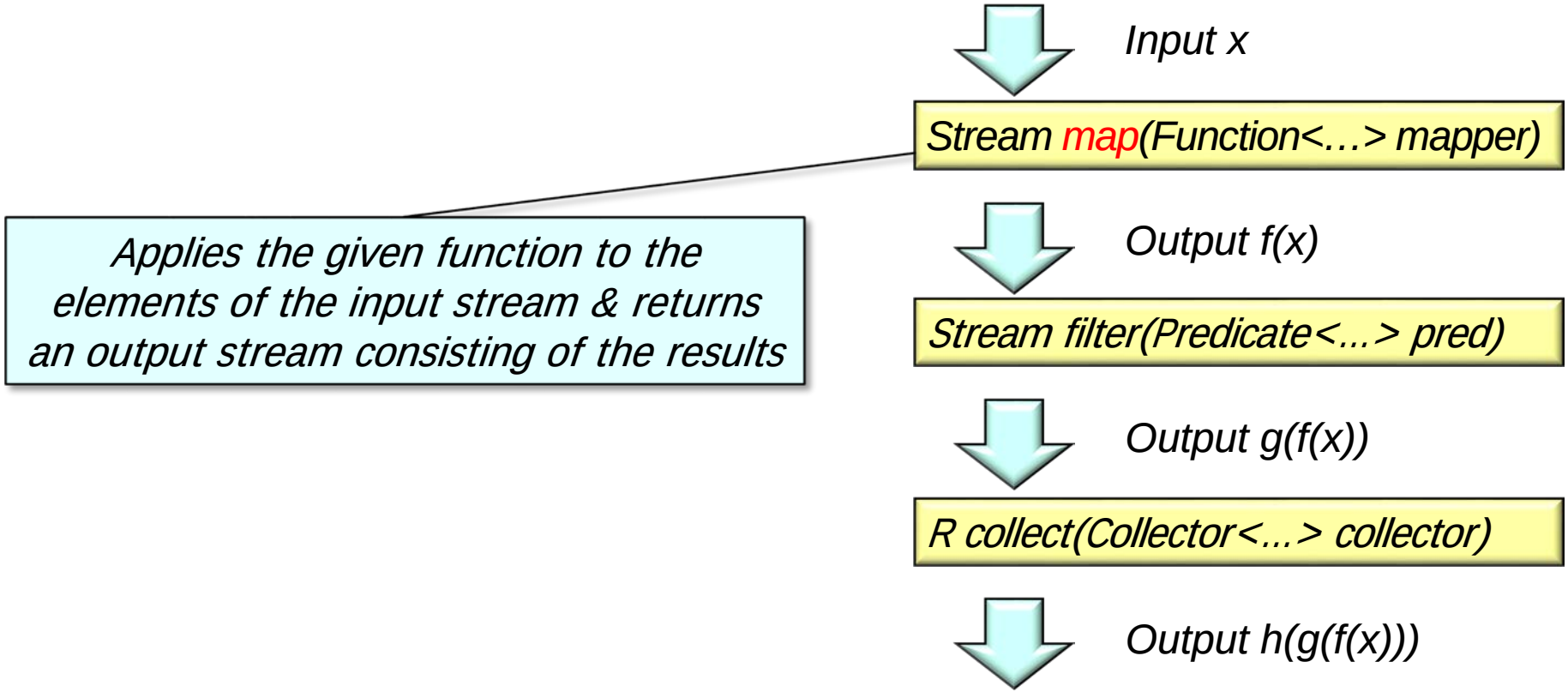
- An aggregate operation processes all elements in a stream



Common aggregate operations include map(), filter(), & collect()

Overview of Common Stream Aggregate Operations

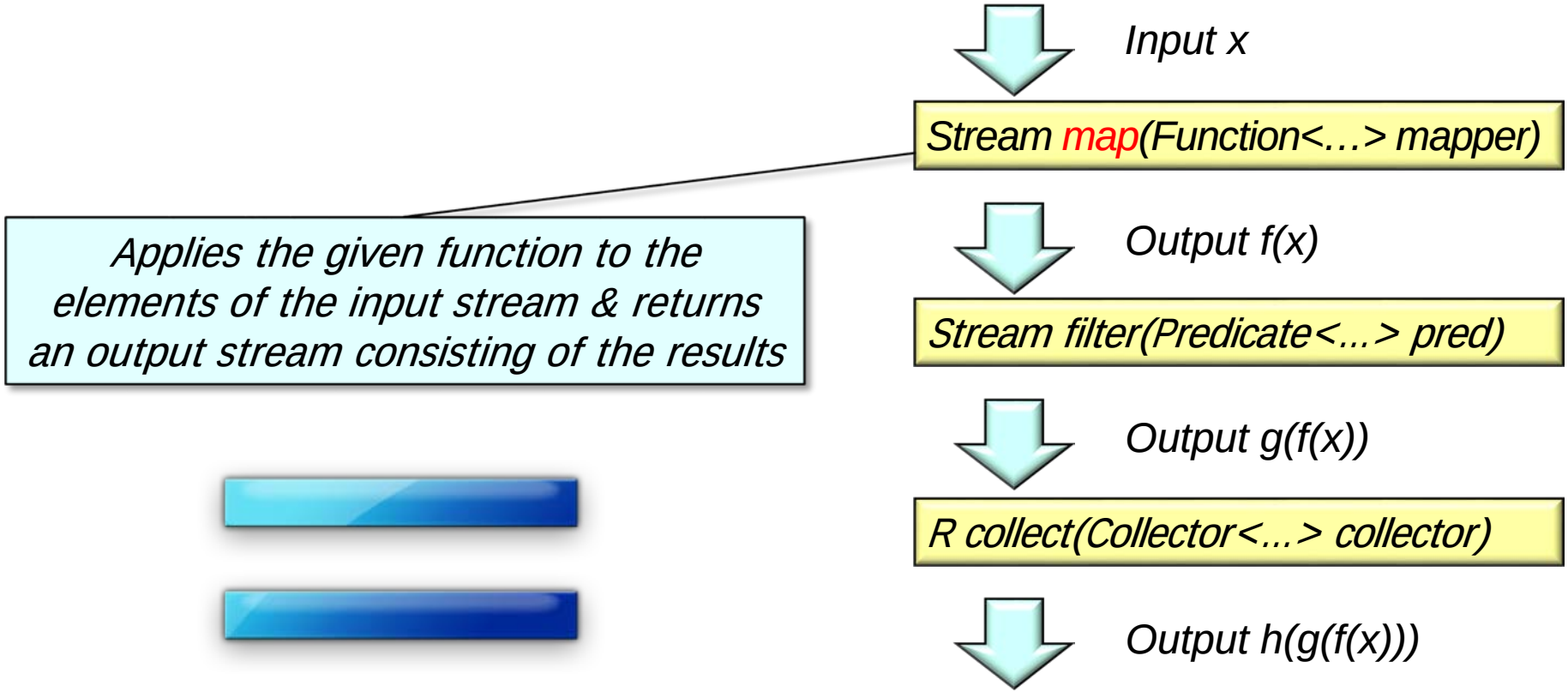
- An aggregate operation processes all elements in a stream



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

Overview of Common Stream Aggregate Operations

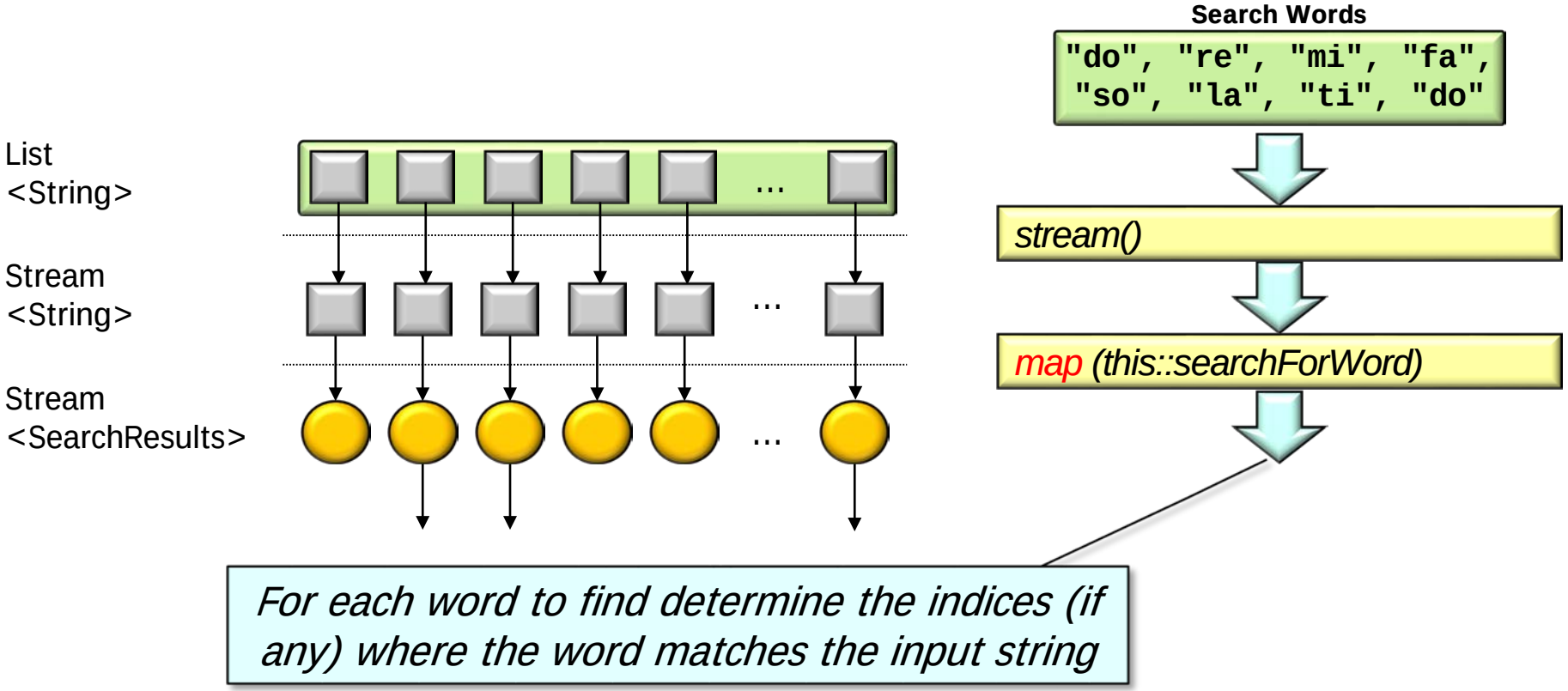
- An aggregate operation processes all elements in a stream



The # of output stream elements matches the # of input stream elements

Overview of Common Stream Aggregate Operations

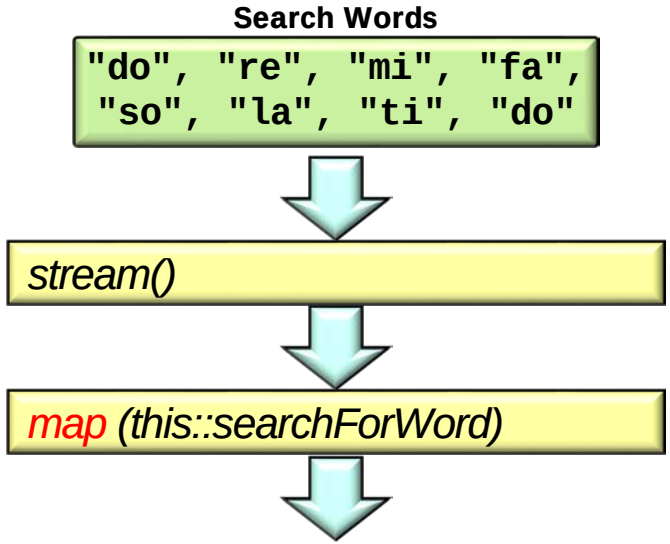
- An aggregate operation processes all elements in a stream



Overview of Common Stream Aggregate Operations

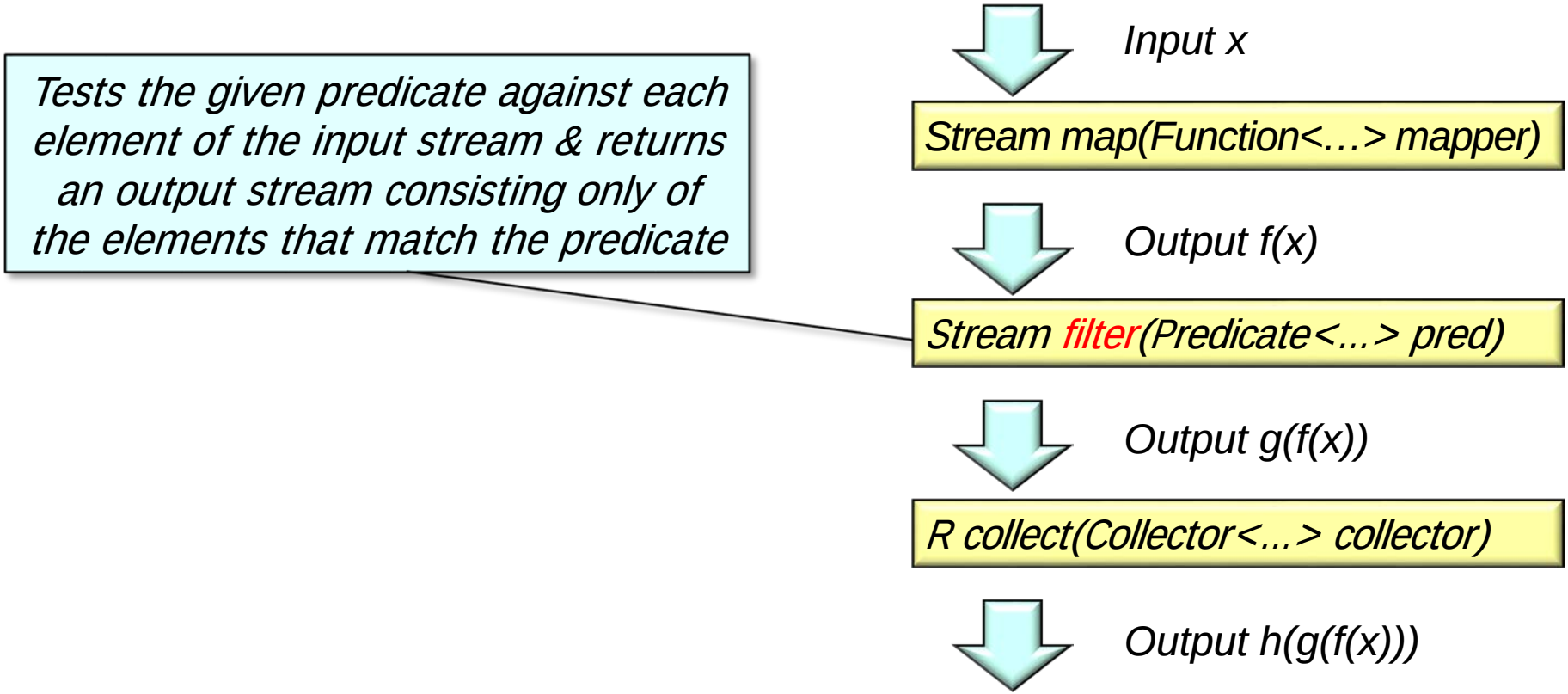
- An aggregate operation processes all elements in a stream

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



Overview of Common Stream Aggregate Operations

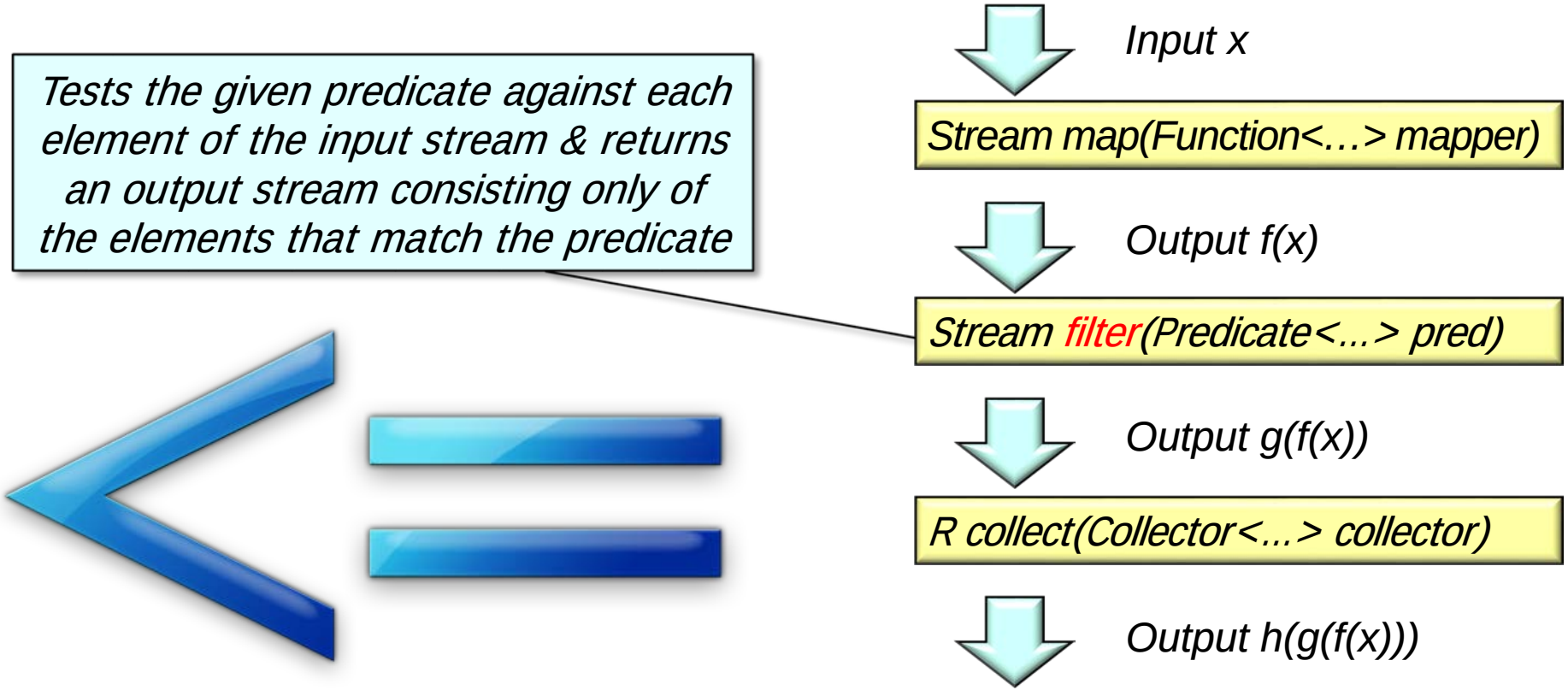
- An aggregate operation processes all elements in a stream



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

Overview of Common Stream Aggregate Operations

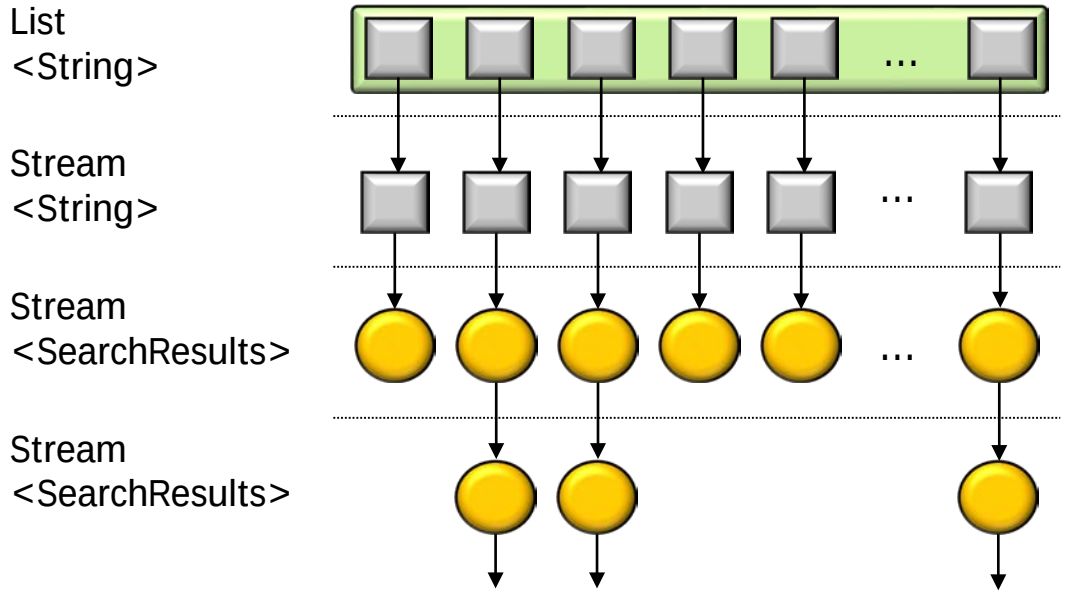
- An aggregate operation processes all elements in a stream

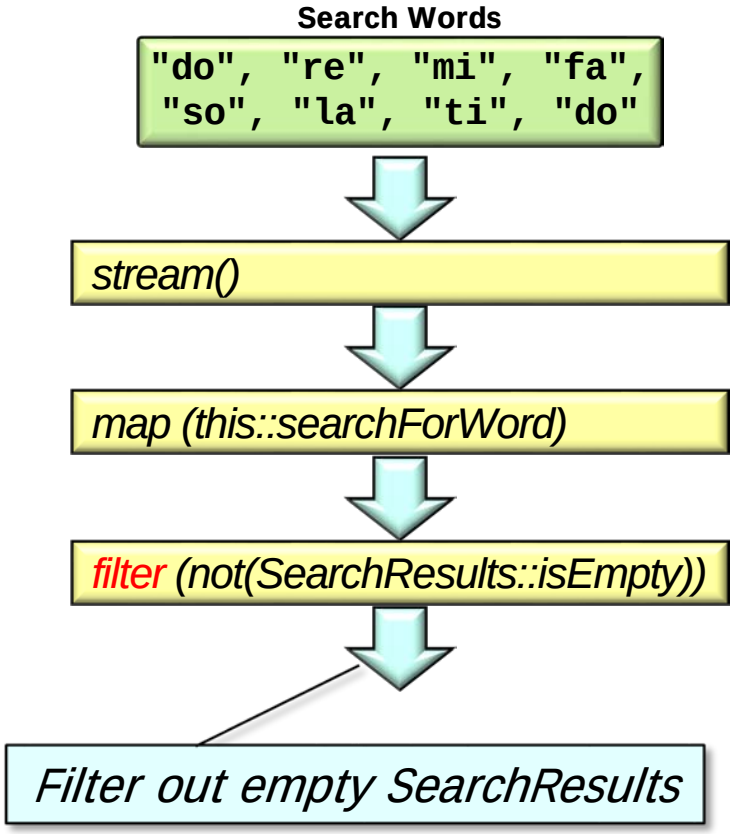


The # of output stream elements may be less than the # of input stream elements

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

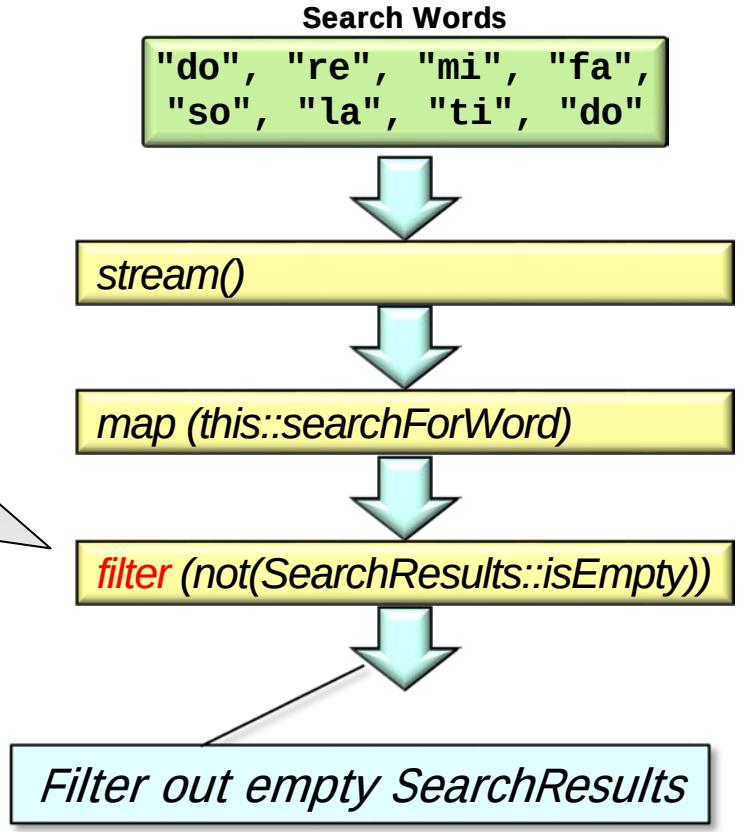




Overview of Common Stream Aggregate Operations

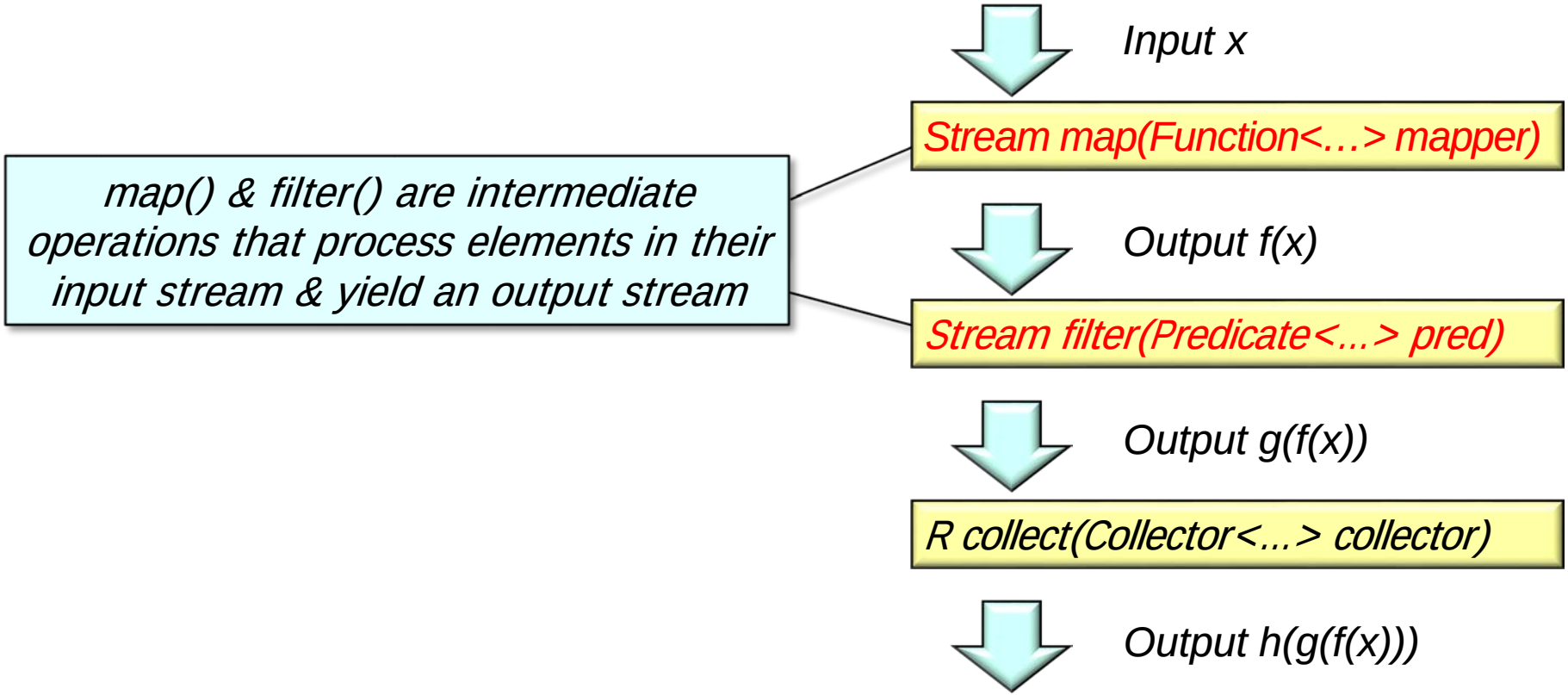
- An aggregate operation processes all elements in a stream

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



Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream



Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<...> pred)

Output g(f(x))

R collect(Collector<...> collector)

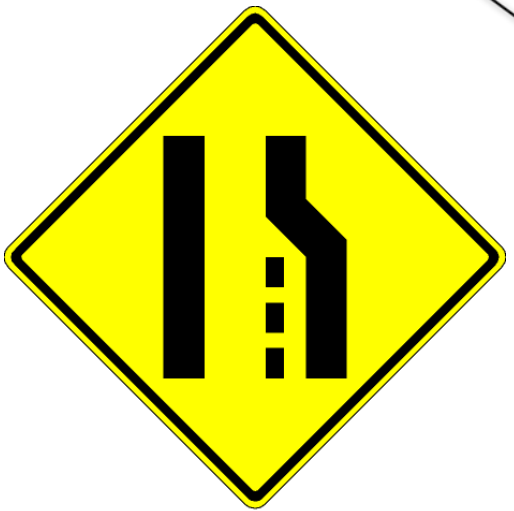
Output h(g(f(x)))

Intermediate operations are “lazy” & don’t run until a terminal operator is reached

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

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))

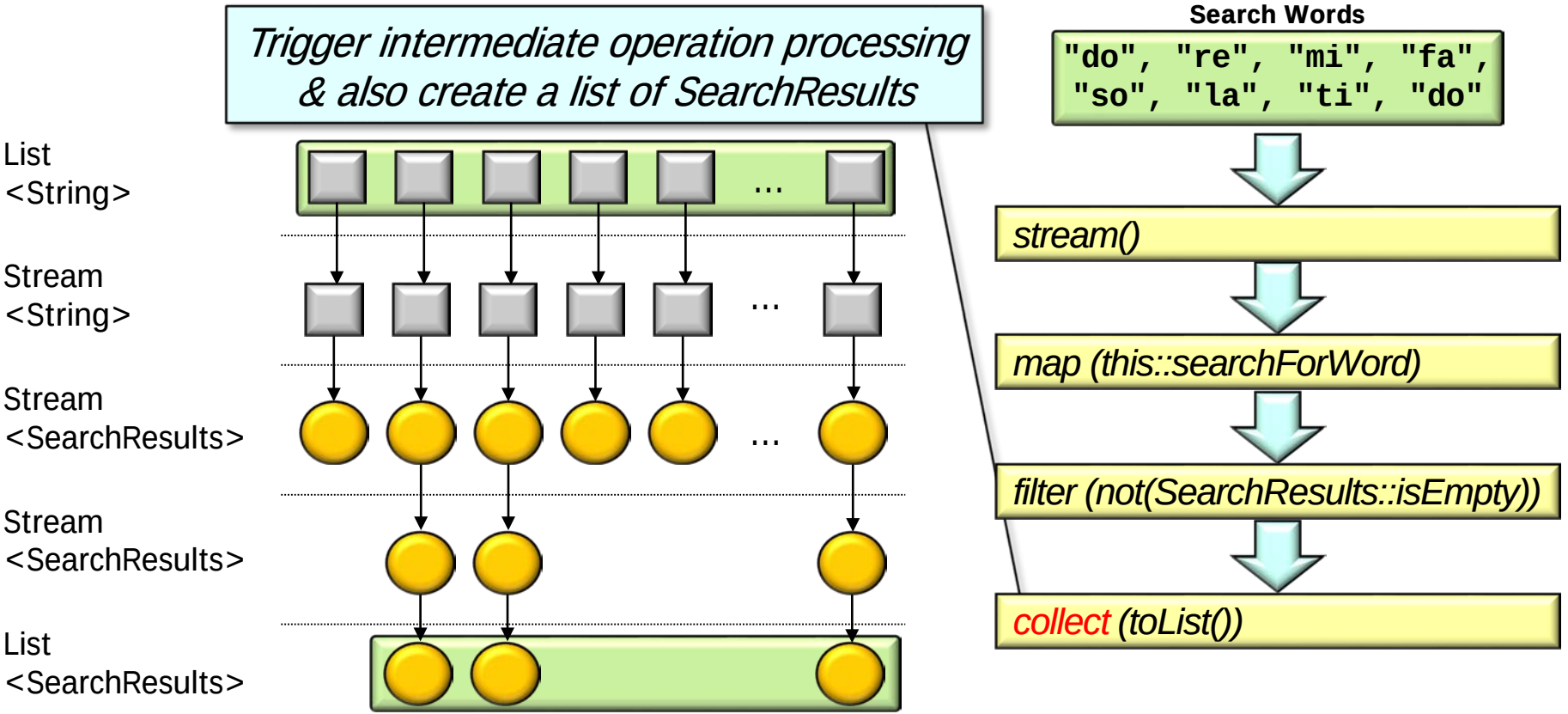
R collect(Collector<…> collector)

Output h(g(f(x)))

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

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

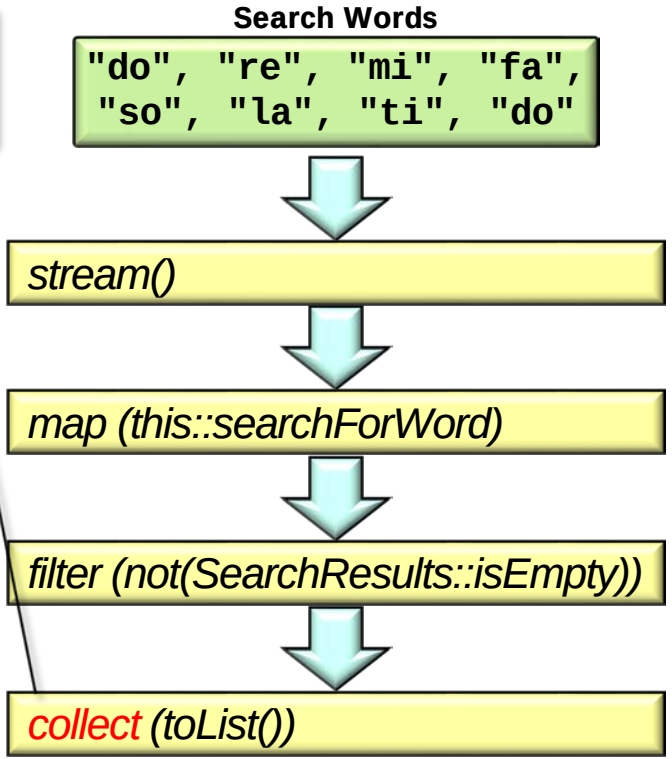


Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

Trigger intermediate operation processing & also create a list of SearchResults

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



End of Overview of Java 8 Streams (Part 2)