

Overview of Java 8 Streams (Part 2)

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

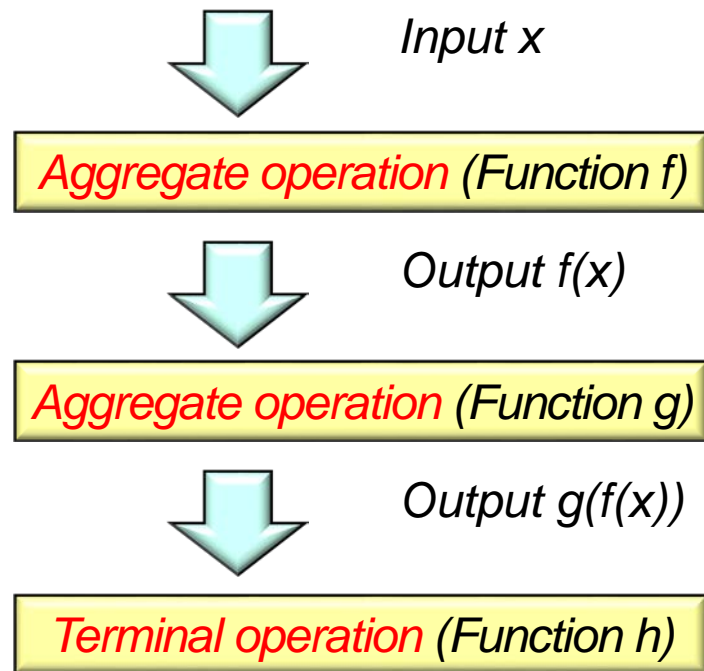
Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA



Learning Objectives in this Part of the Lesson

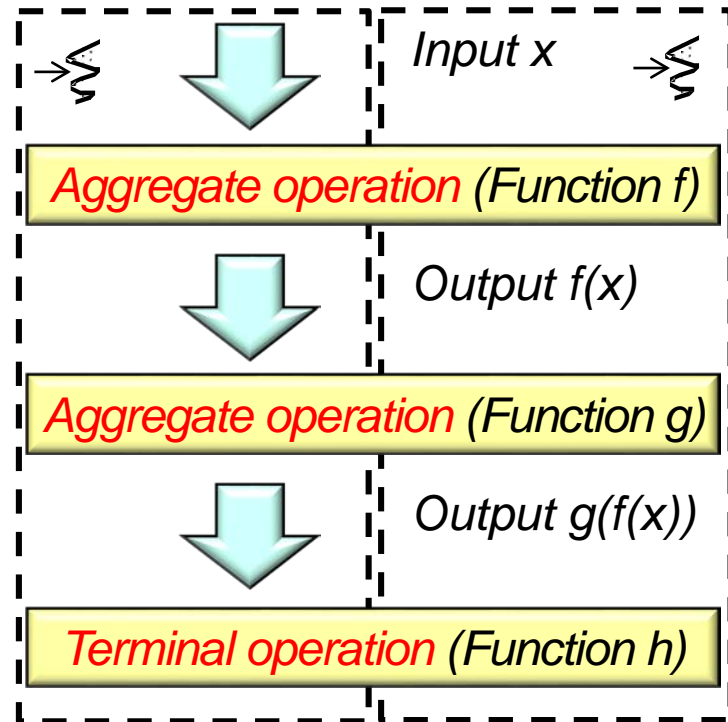
- Understand the structure & functionality of Java 8 streams, e.g.,
 - Fundamentals of streams
 - Common stream aggregate operations



Learning Objectives in this Part of the Lesson

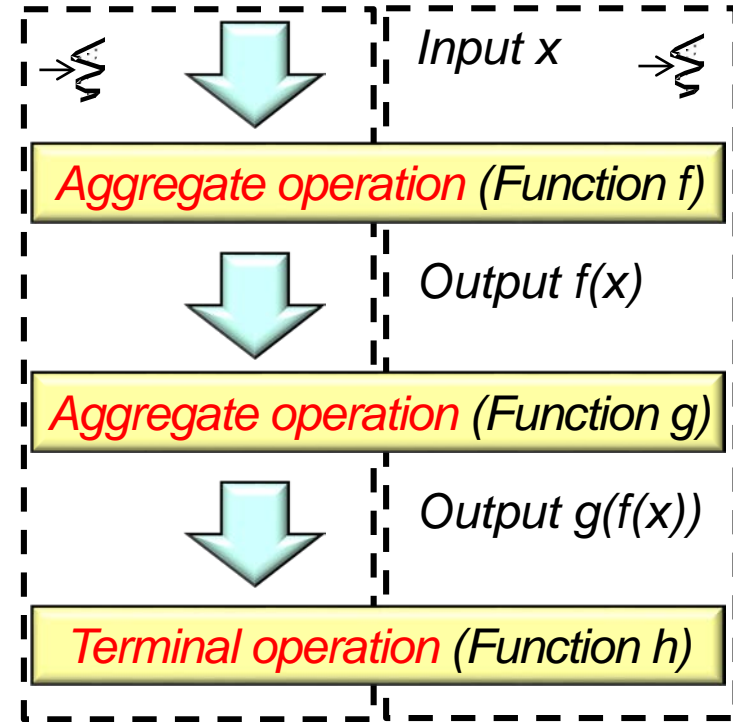
- Understand the structure & functionality of Java 8 streams, e.g.,

- 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

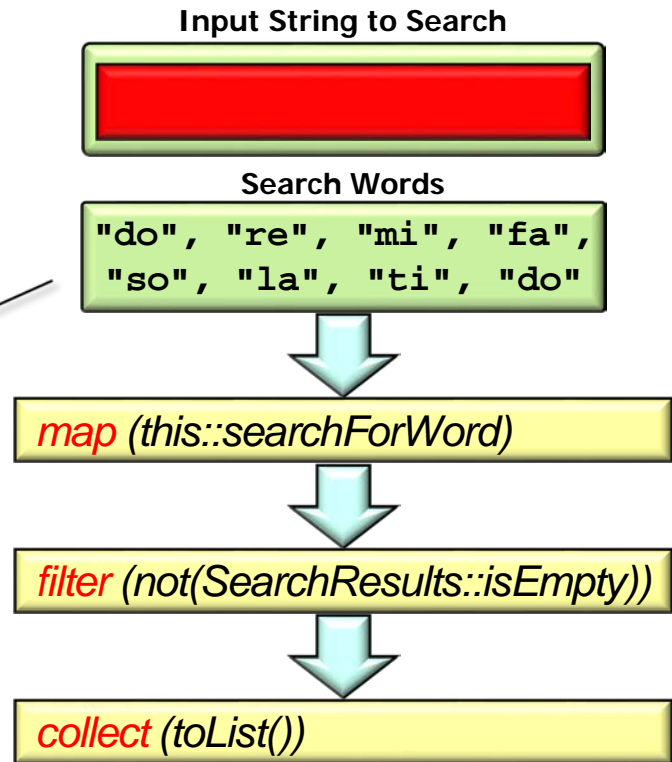


Being a good streams programmer makes you a better parallel streams programmer

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/douglasraigschmidt/LiveLessons/tree/master/SimpleSearchStream

Overview of SimpleSearch Stream Example

Overview of SimpleSearchStream Example

- This example finds words in an input string

Input String to Search

Let's start at the very beginning...

*We'll use this example to explain
Java 8 aggregate operations
throughout this part of the lesson*

Search Words

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

stream()

map (this::searchForWord)

filter (not(SearchResults::isEmpty))

collect (toList())

Overview of SimpleSearchStream Example

- This example finds words in an input string

Input String to Search

Let's start at the very beginning...

Search Words

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

stream()

map (this::searchForWord)

filter (not(SearchResults::isEmpty))

collect (toList())

Starting SimpleSearchStream

Word "Re" matched at index [131|141|151|202|212|222|
979|1025|1219|1259|
1278|1300|1351|1370|1835|
1875|1899|1939|2266|2295]

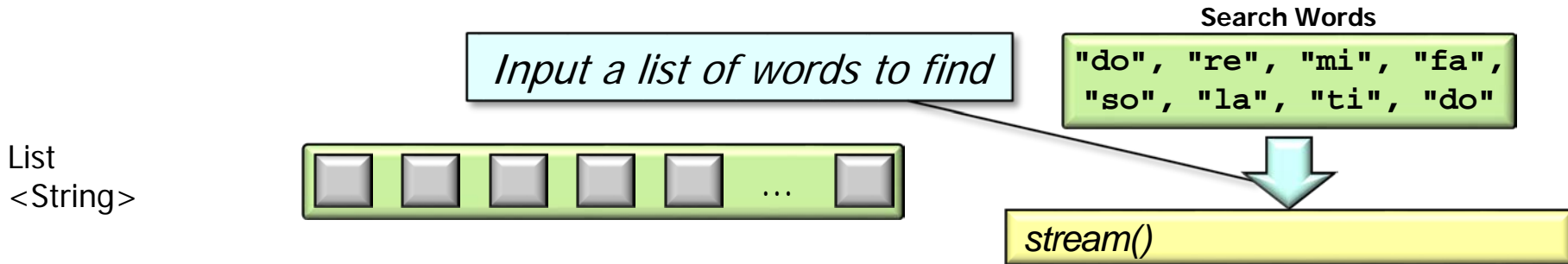
Word "Ti" matched at index [237|994|1272|1294|1364|1850|
1860|1912|1915|1952|1955|
2299]

Word "La" matched at index [234|417|658|886|991|1207|
1247|1269|1291|1339|1361|
1742|1847|1863|1909|1949|
2161|2254|2276|2283]...

Ending 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"

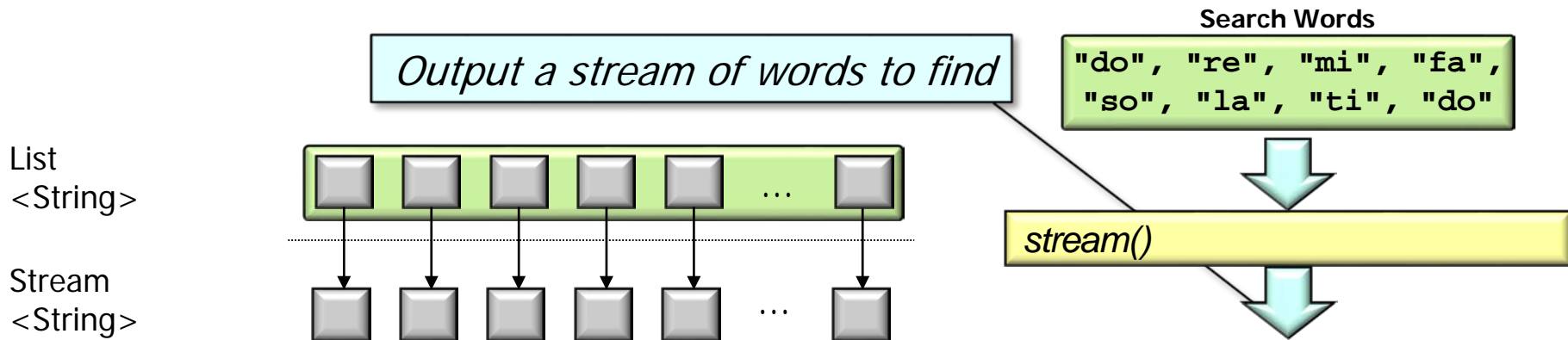


stream()

Convert collection to a (sequential) stream

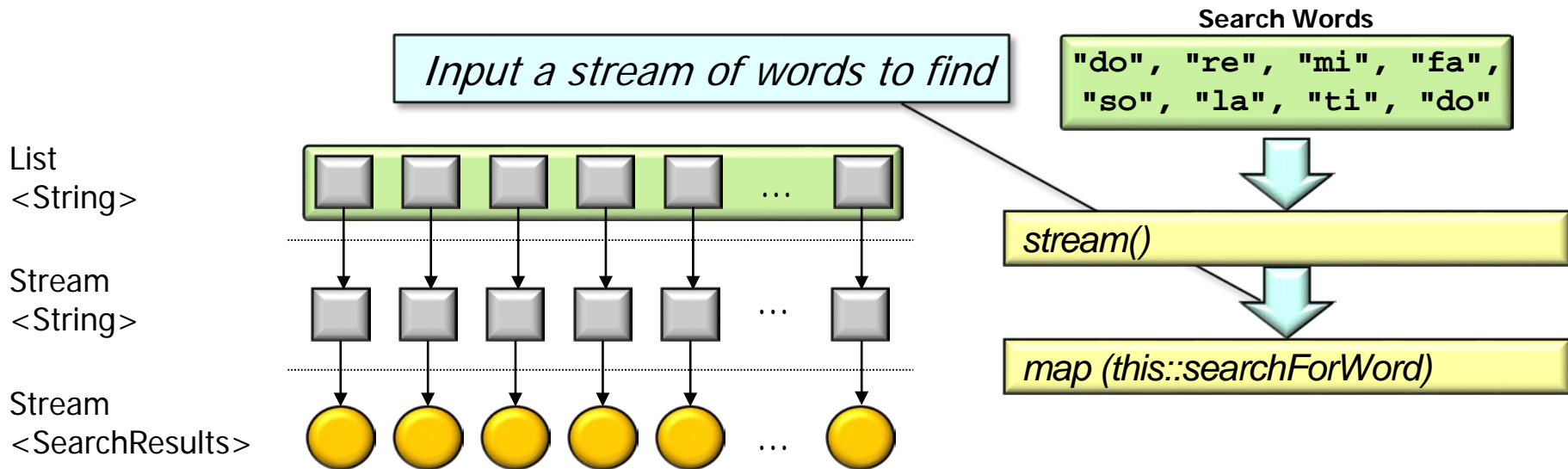
Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string



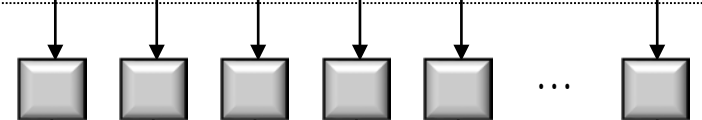
Overview of SimpleSearchStream Example

- This example finds words in an input string

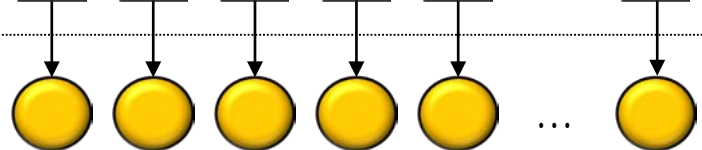
List
<String>



Stream
<String>



Stream
<SearchResults>



Search Words

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



stream()



map (*this::searchForWord*)

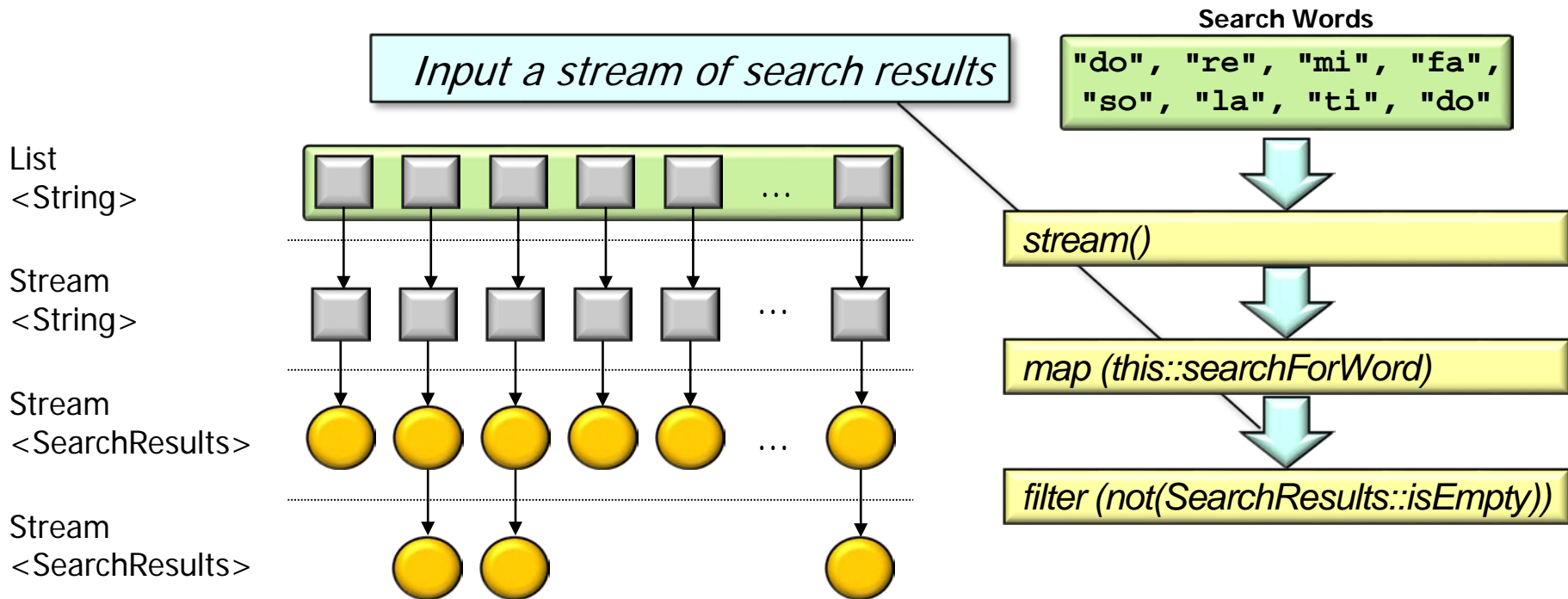
Search for the word in each input string

- This example finds words in an input string



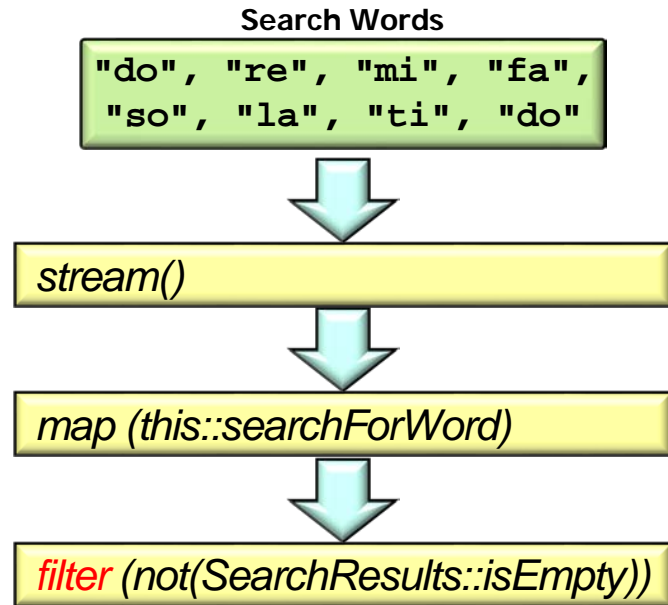
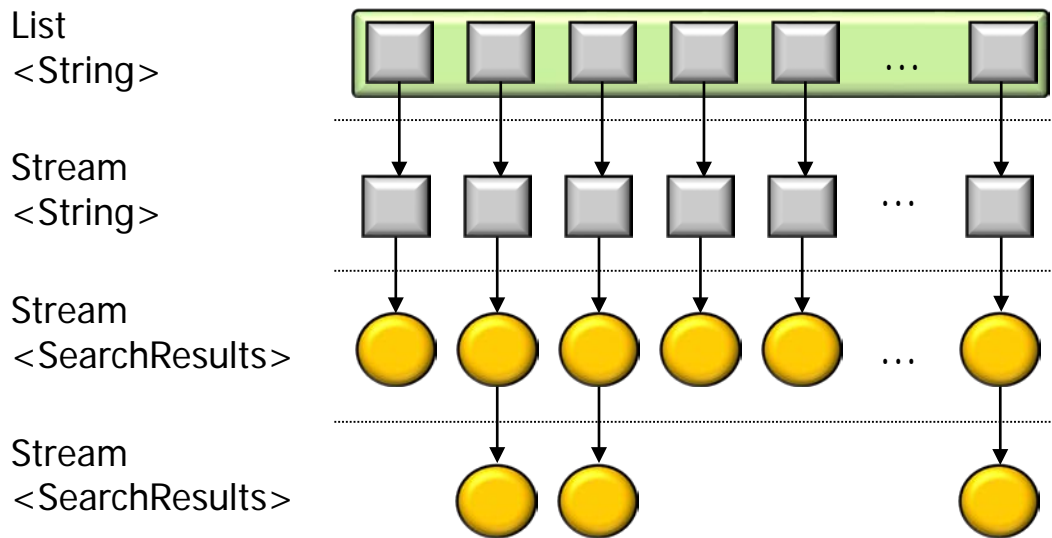
Overview of SimpleSearchStream Example

- This example finds words in an input string



Overview of SimpleSearchStream Example

- This example finds words in an input string



Remove empty search results from the stream

Overview of SimpleSearchStream Example

- This example finds words in an input string

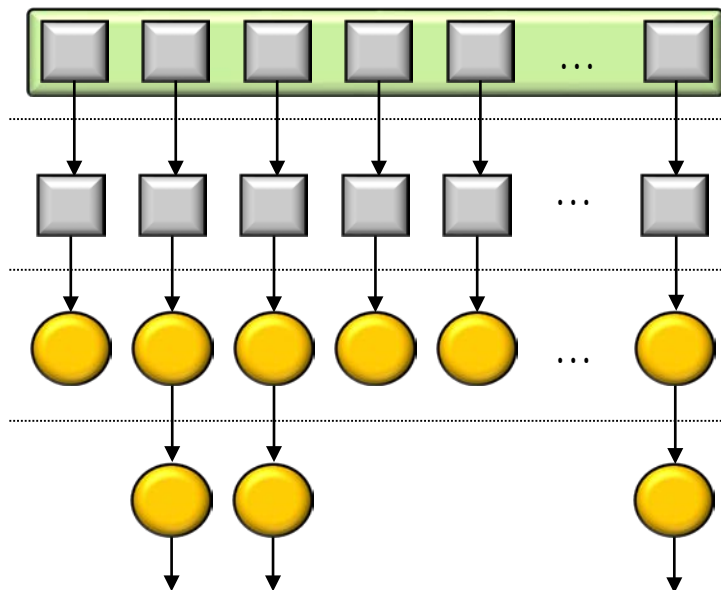
Output a stream of non-empty search results

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))`

Overview of SimpleSearchStream Example

- This example finds words in an input string

Input a stream of non-empty search results

List
<String>



Stream
<String>



Stream
<SearchResults>



Stream
<SearchResults>



List
<SearchResults>



Search Words

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

`stream()`

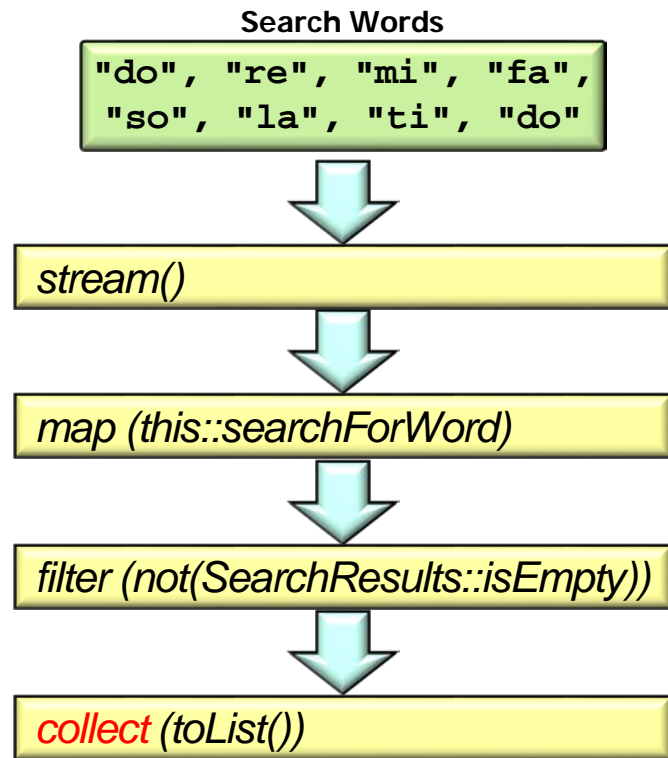
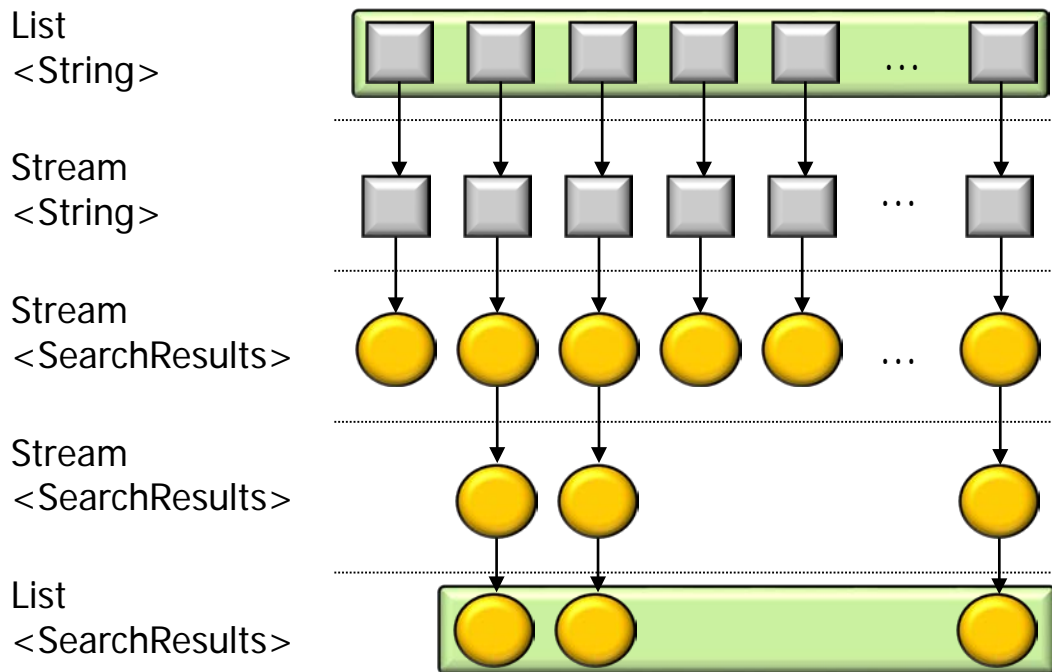
`map (this::searchForWord)`

`filter (not(SearchResults::isEmpty))`

`collect (toList())`

Overview of SimpleSearchStream Example

- This example finds words in an input string

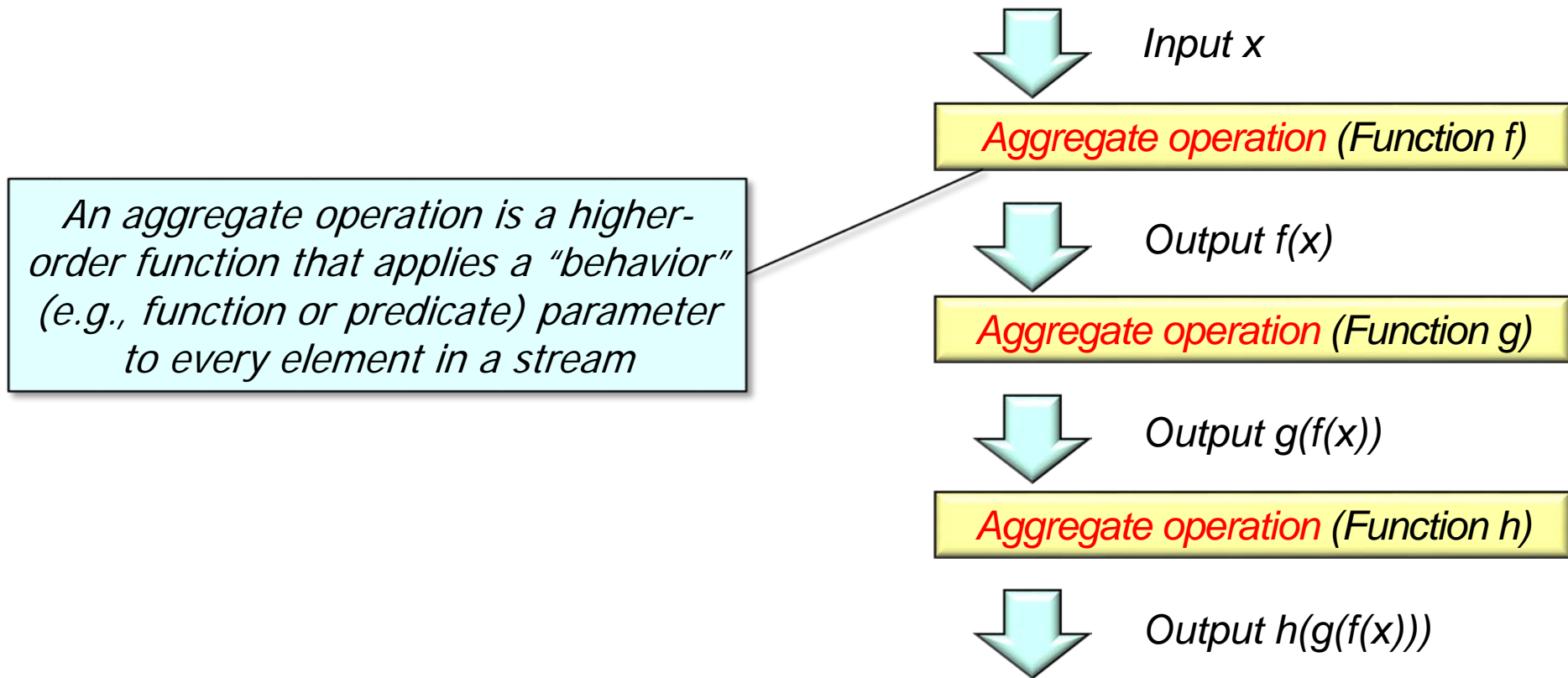


Trigger intermediate operation processing & return a list of SearchResults

Overview of Common Stream Aggregate Operations

Overview of Common Stream Aggregate Operations

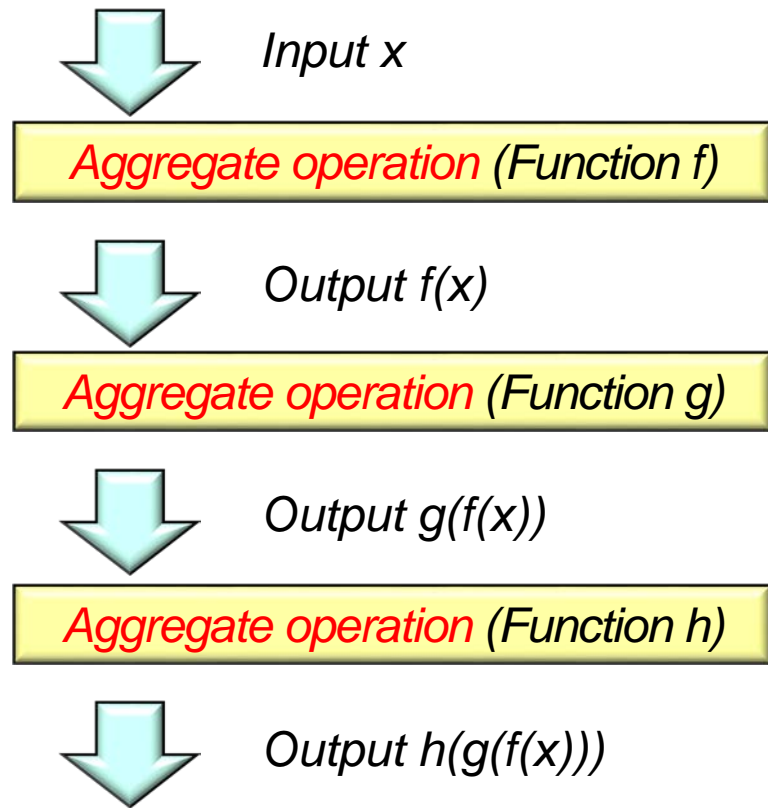
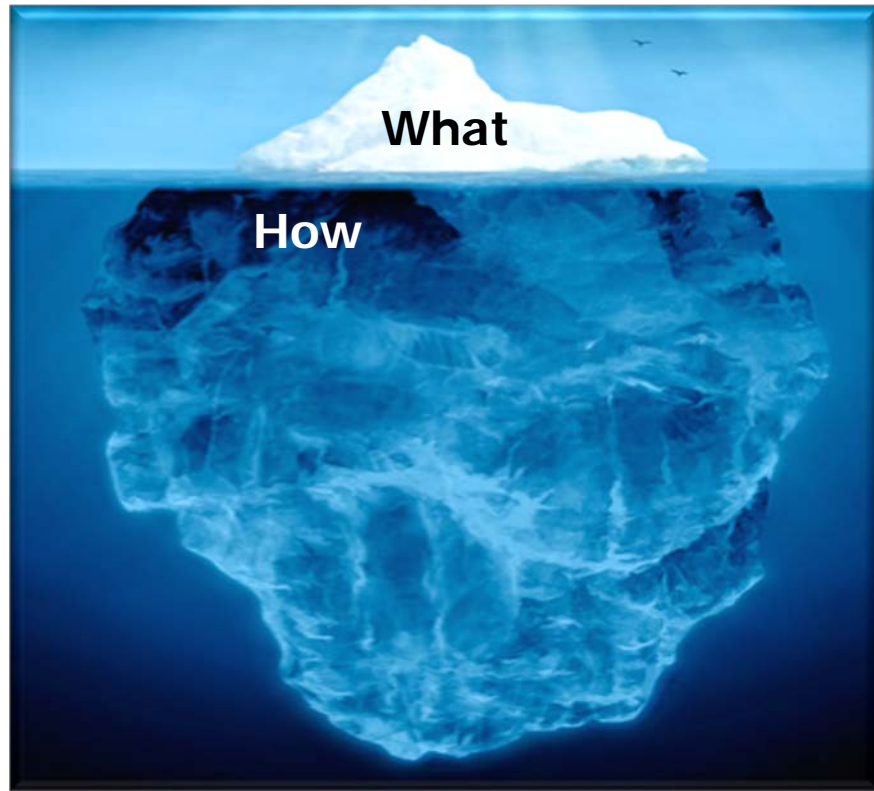
- An aggregate operation processes all elements in a stream



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



Input x

Stream **map**(Function<...> mapper)



Output $f(x)$

Stream **filter**(Predicate<...> pred)



Output $g(f(x))$

R **collect**(Collector<...> collector)

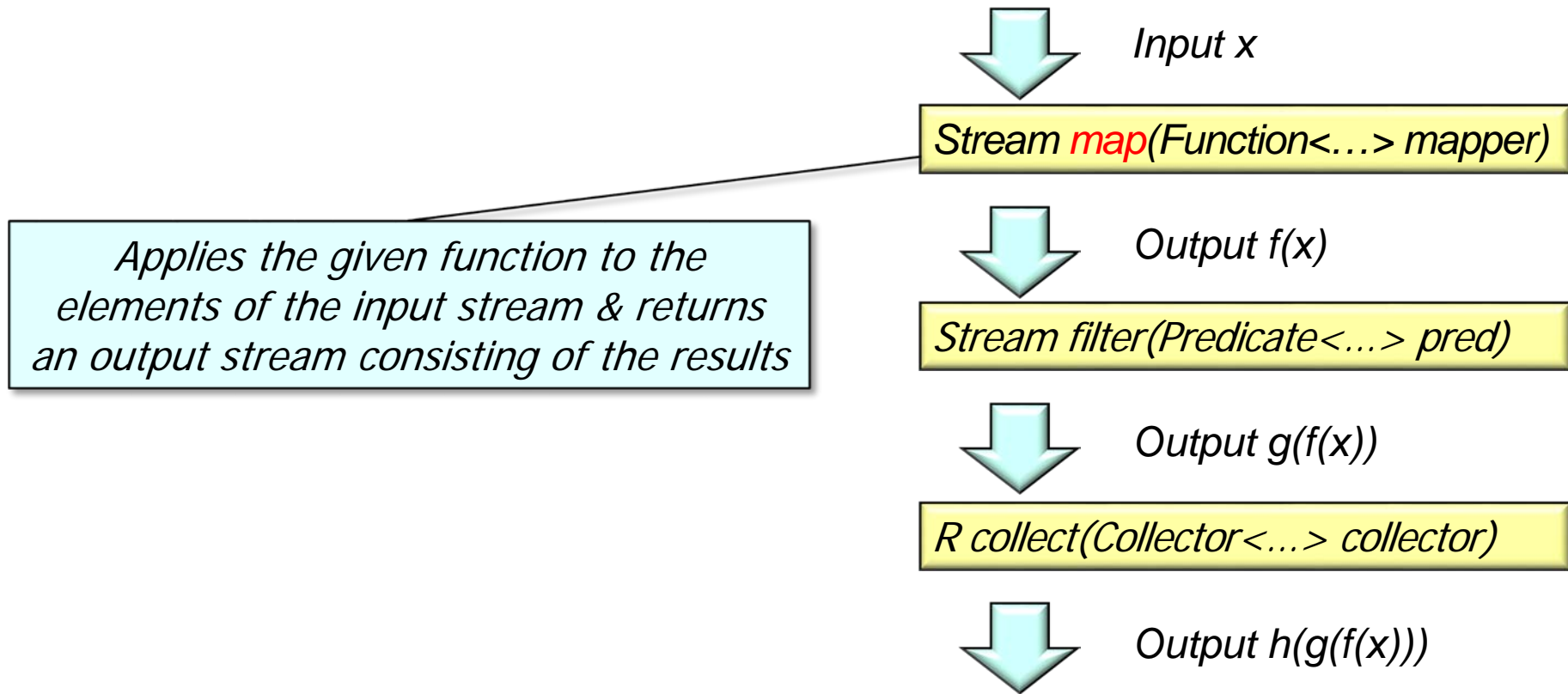


Output $h(g(f(x)))$

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

Overview of Common Stream Aggregate Operations

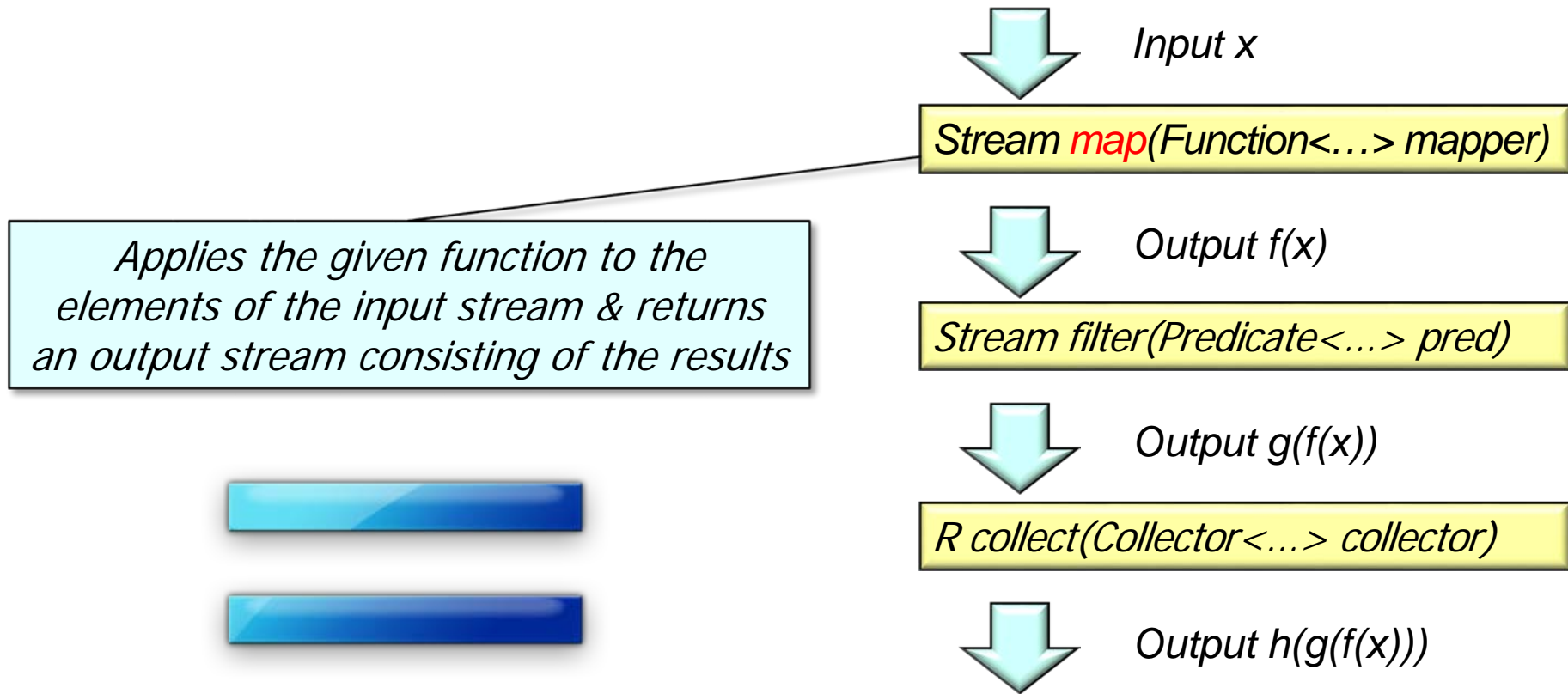
- An aggregate operation processes all elements in a stream



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

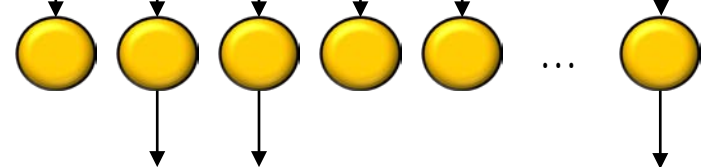
List
<String>



Stream
<String>



Stream
<SearchResults>



For each word to find determine the indices (if any) where the word matches the input string

Search Words

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

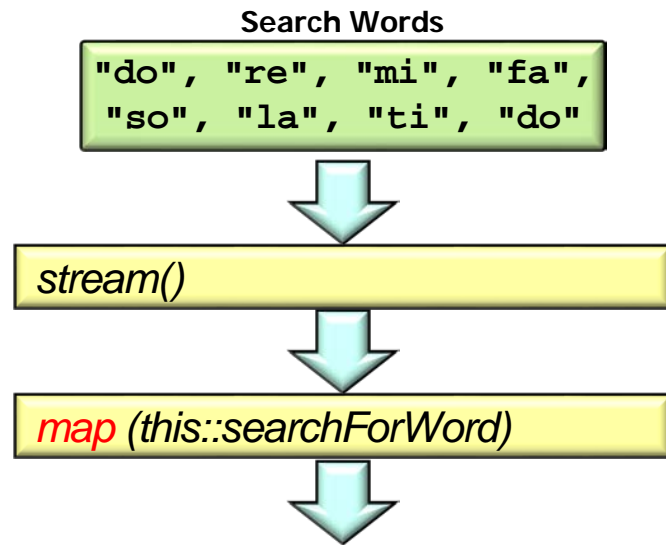
`stream()`

`map (this::searchForWord)`

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

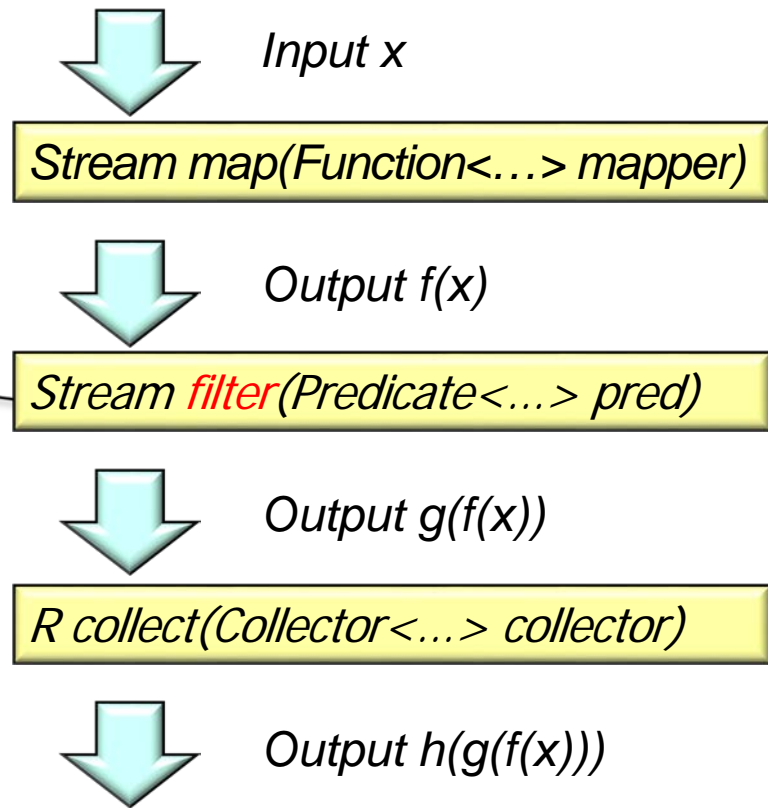
```
List<SearchResults> results =  
    wordsToFind  
        .stream()  
        .map(this::searchForWord)  
        .filter(not  
            (SearchResults::isEmpty))  
        .collect(toList());
```



Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

Tests the given predicate against each element of the input stream & returns an output stream consisting only of the elements that match the predicate

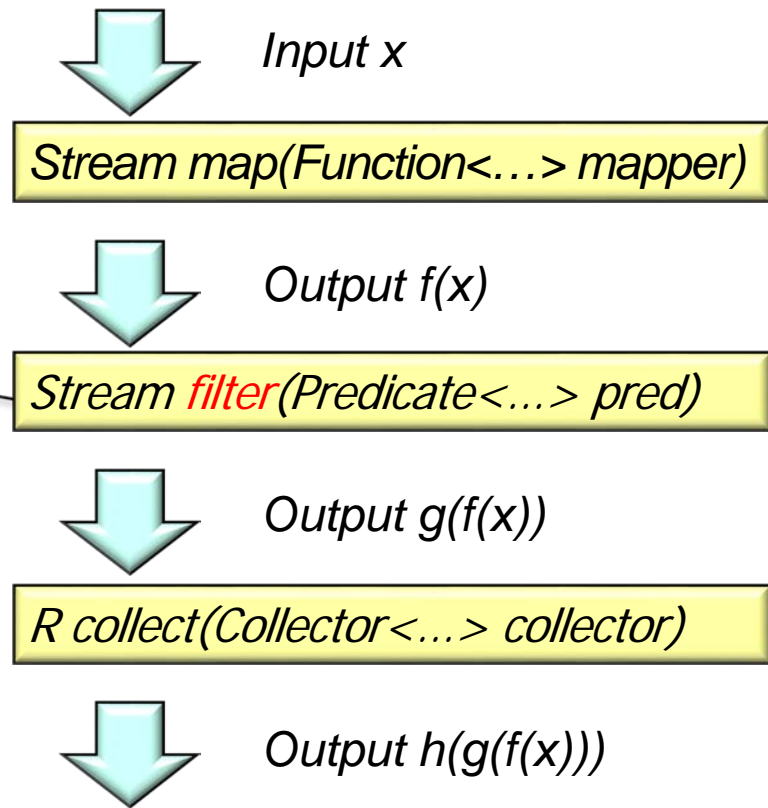
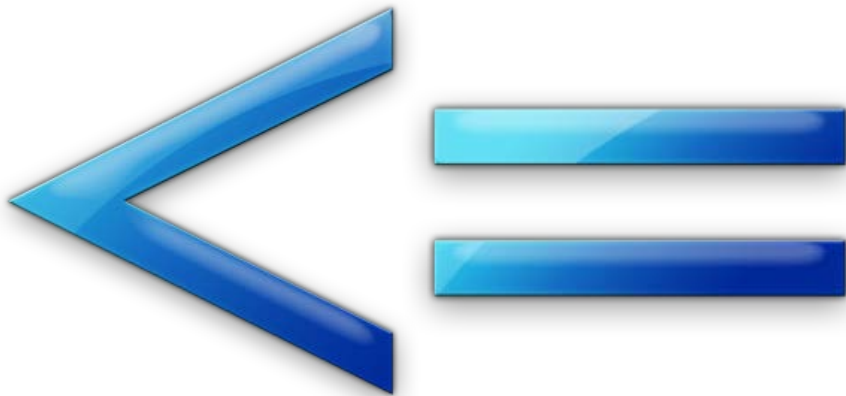


See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#filter

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

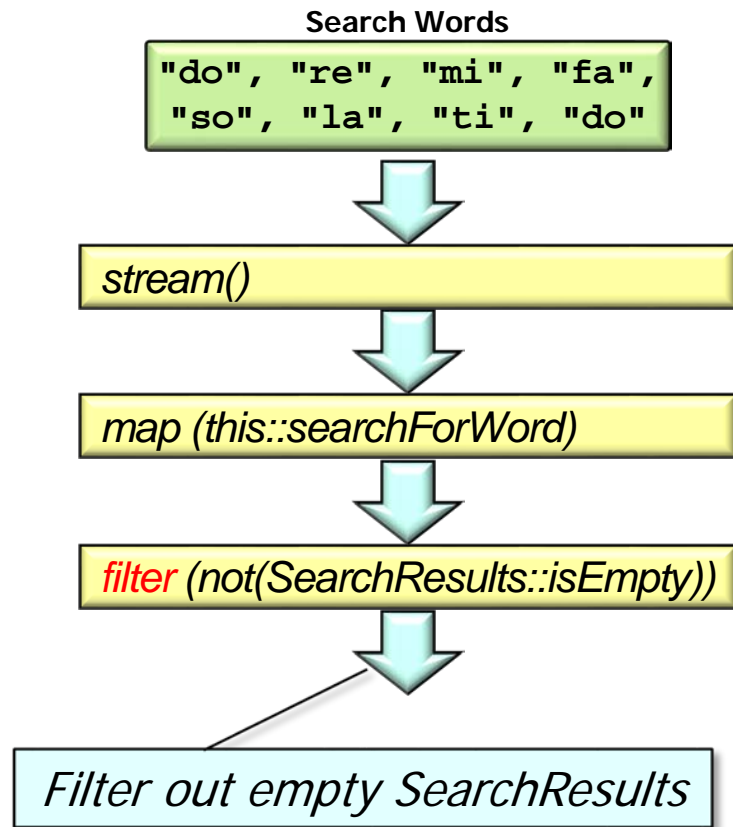
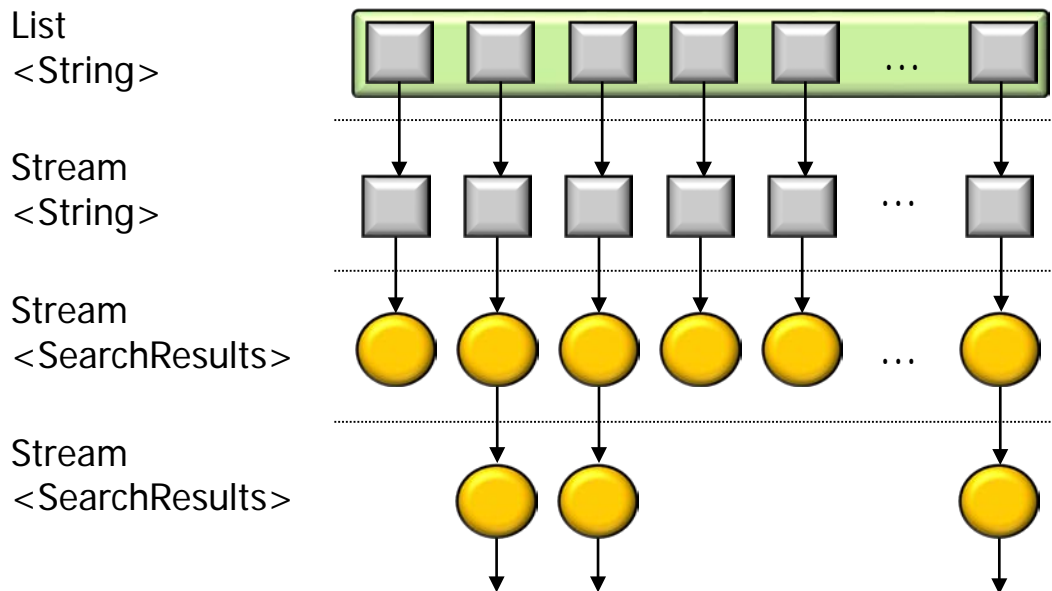
Tests the given predicate against each element of the input stream & returns an output stream consisting only of the elements that match the predicate



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

Overview of Common Stream Aggregate Operations

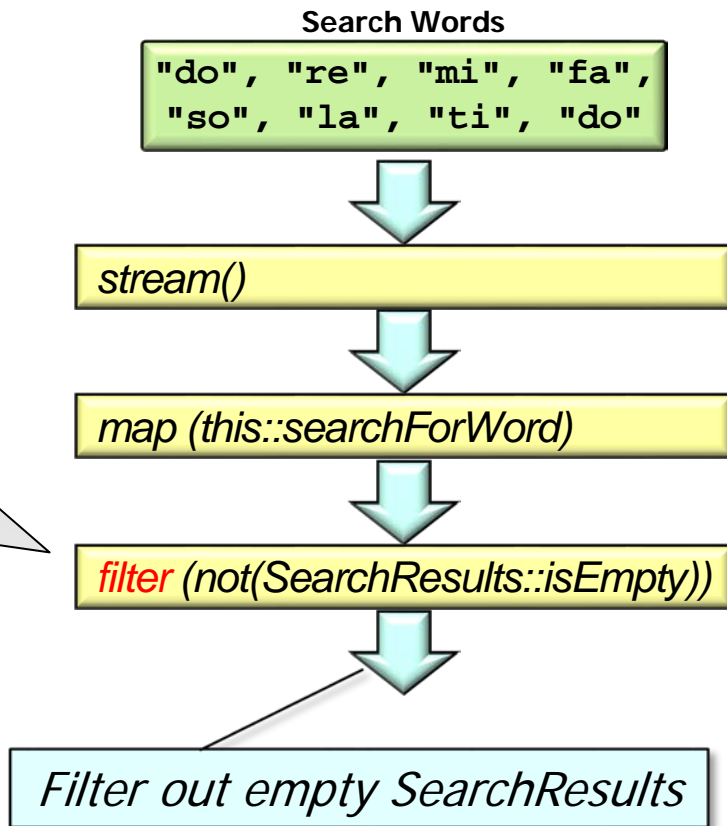
- An aggregate operation processes all elements in a stream



Overview of Common Stream Aggregate Operations

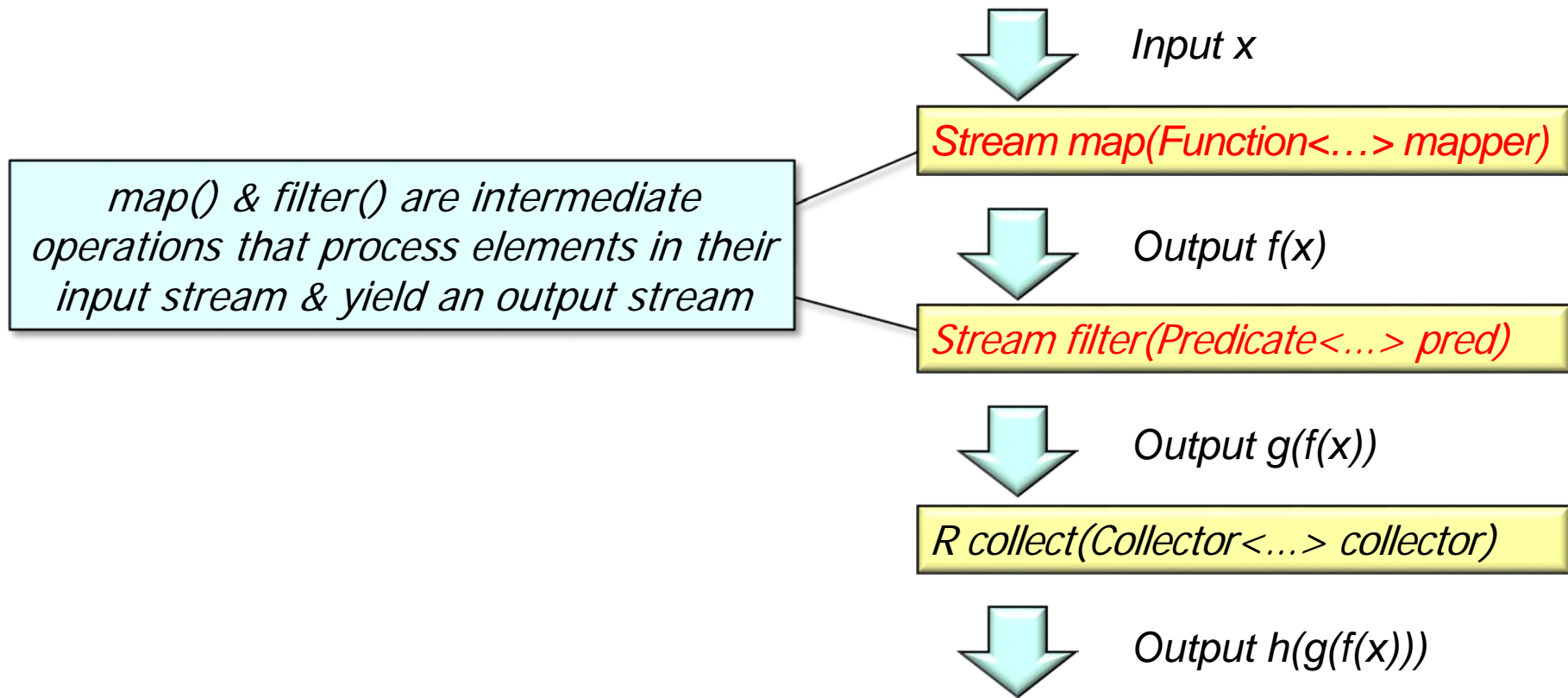
- An aggregate operation processes all elements in a stream

```
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)



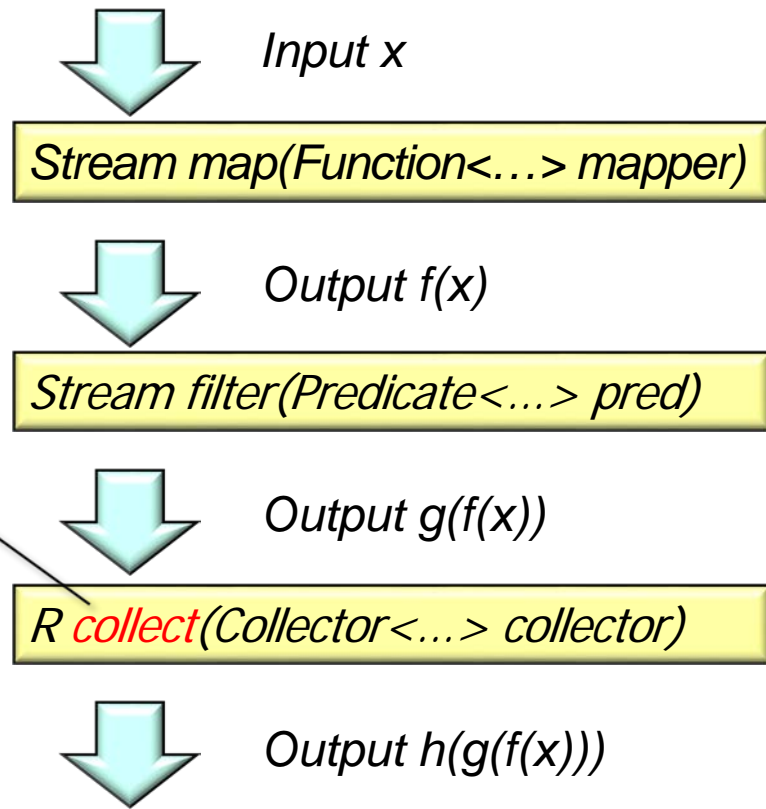
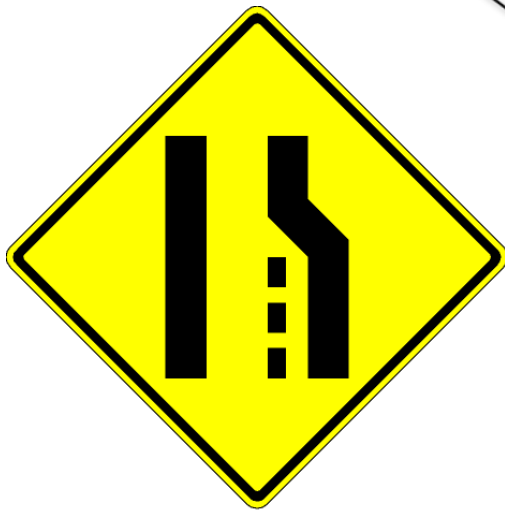
Output $h(g(f(x)))$

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



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

*Trigger intermediate operation processing
& also create a list of SearchResults*

List
<String>



Stream
<String>



Stream
<SearchResults>



Stream
<SearchResults>



List
<SearchResults>



Search Words

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

`stream()`

`map (this::searchForWord)`

`filter (not(SearchResults::isEmpty))`

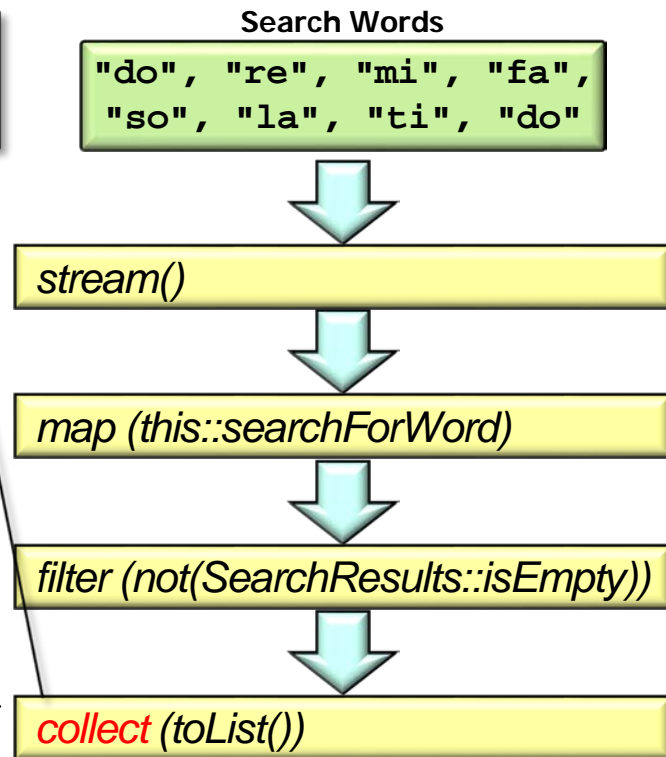
`collect (toList())`

Overview of Common Stream Aggregate Operations

- An aggregate operation processes all elements in a stream

*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)