Java Streams: Intermediate Operations

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

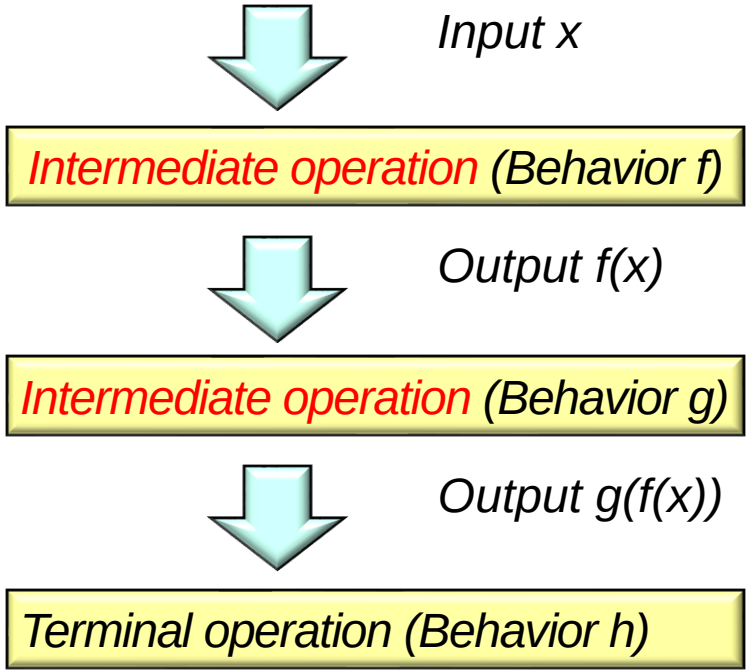
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

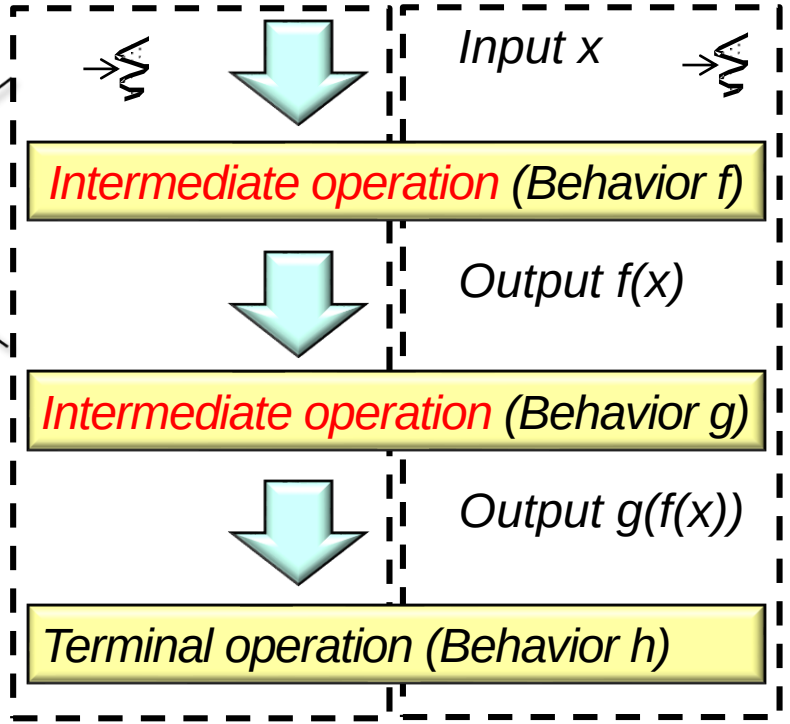
- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations



Learning Objectives in this Part of the Lesson

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

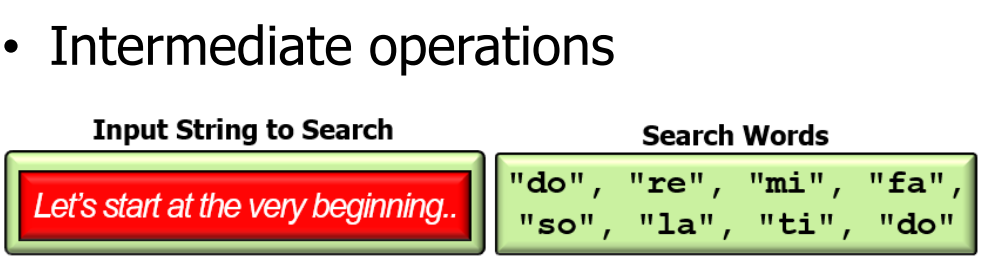
These operations apply to both sequential & parallel streams



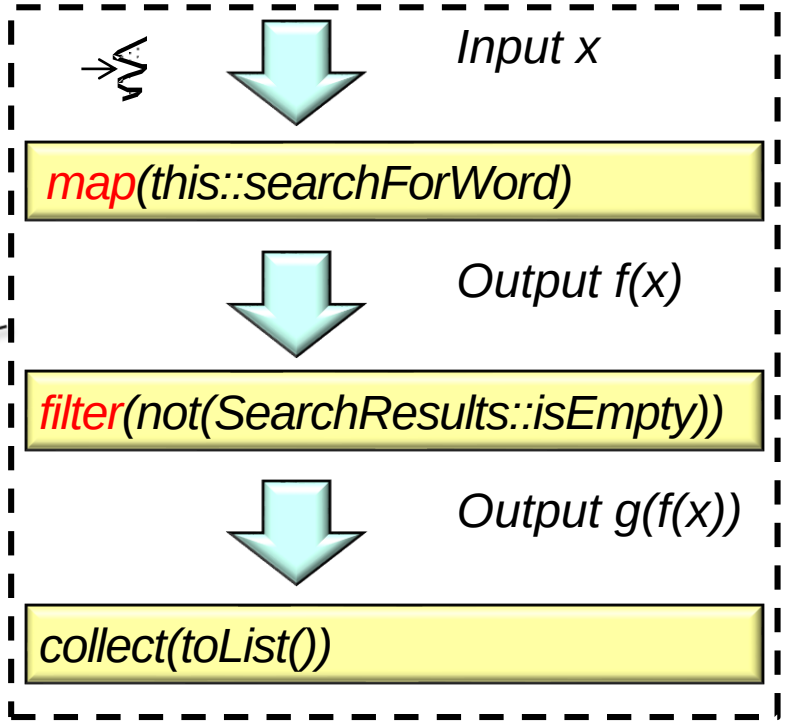
Being a good streams programmer makes you a better parallel streams programmer

Learning Objectives in this Part of the Lesson

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

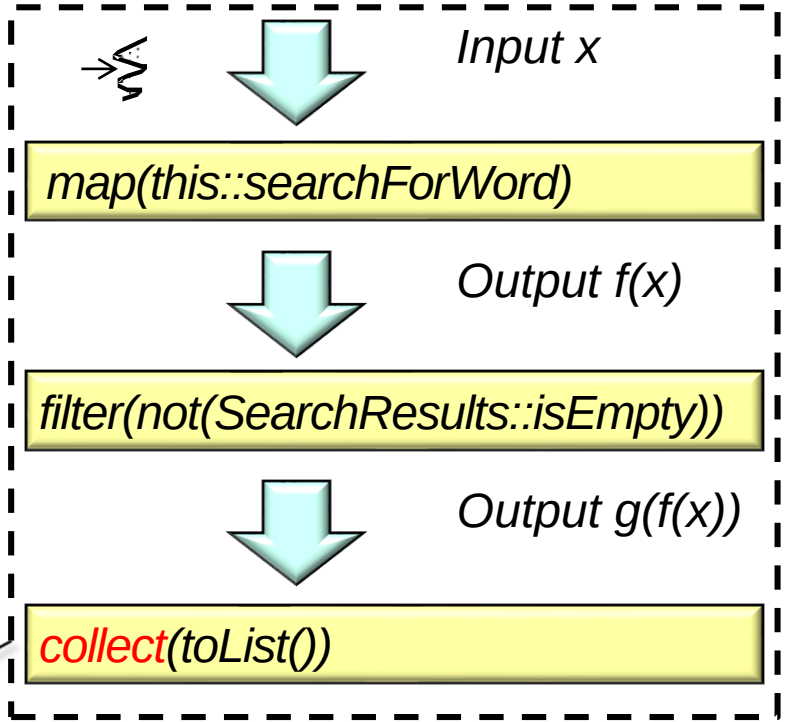


We continue to showcase the SimpleSearchStream program



Learning Objectives in this Part of the Lesson

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



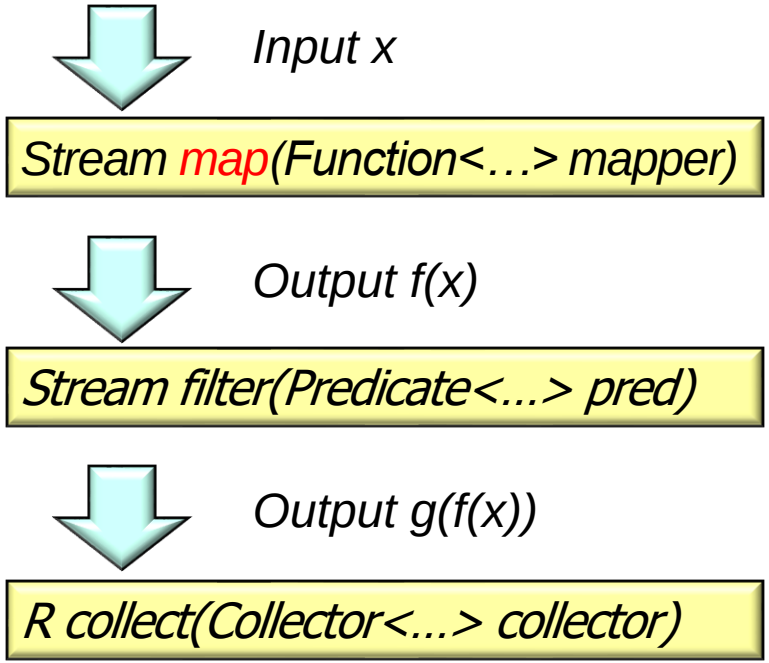
Intermediate operations are "lazy" & run only after terminal operator is reached.

See www.logicbig.com/tutorials/core-java-tutorial/java-util-stream/lazy-evaluation

Overview of the map() Intermediate Operation

Overview of the map() Intermediate Operation

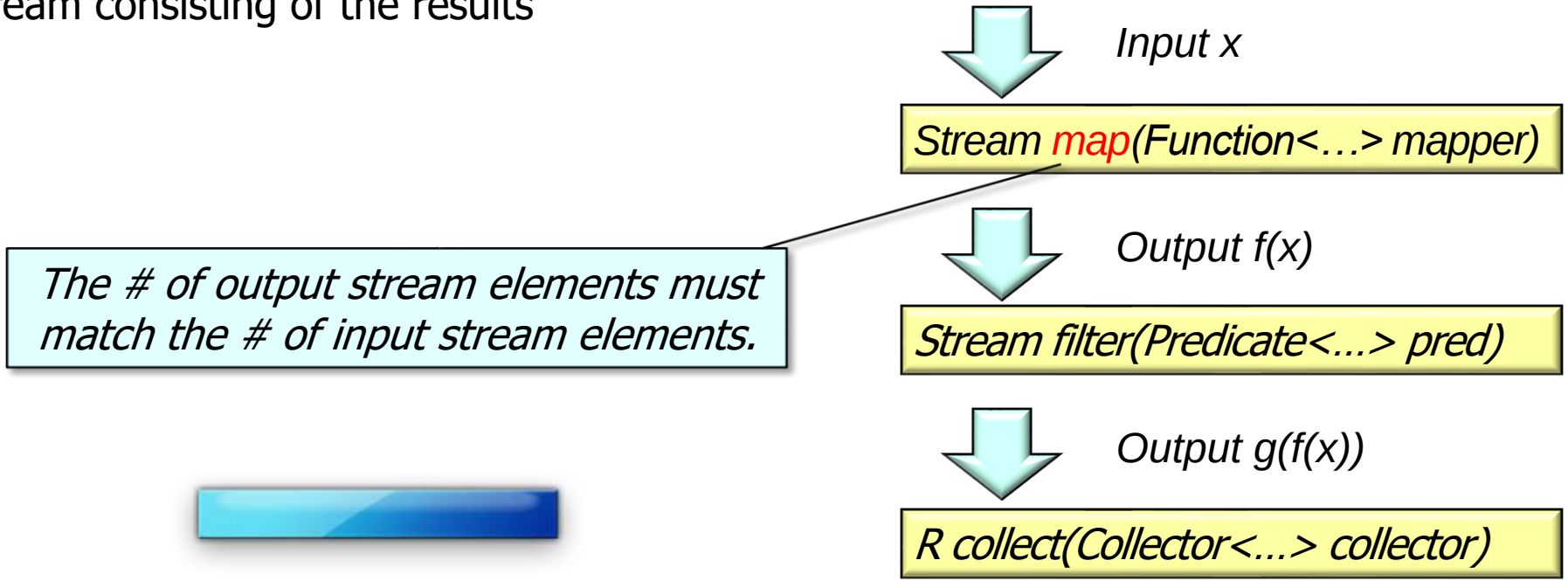
- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results



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

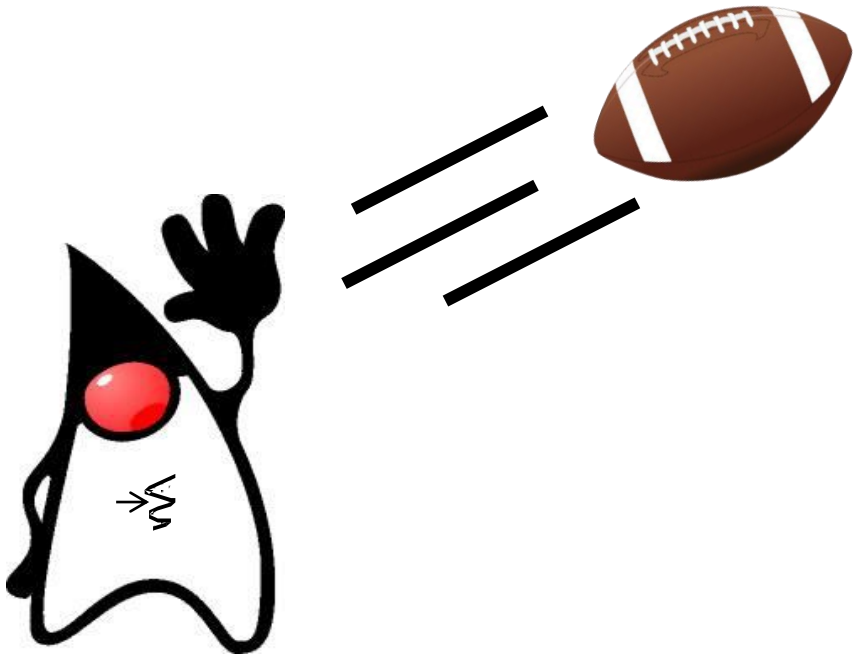
Overview of the map() Intermediate Operation

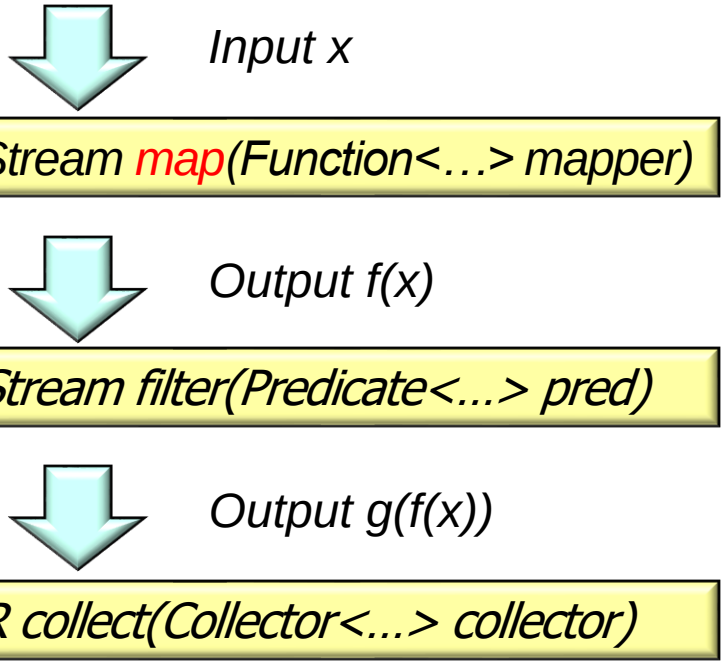
- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results



Overview of the map() Intermediate Operation

- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results

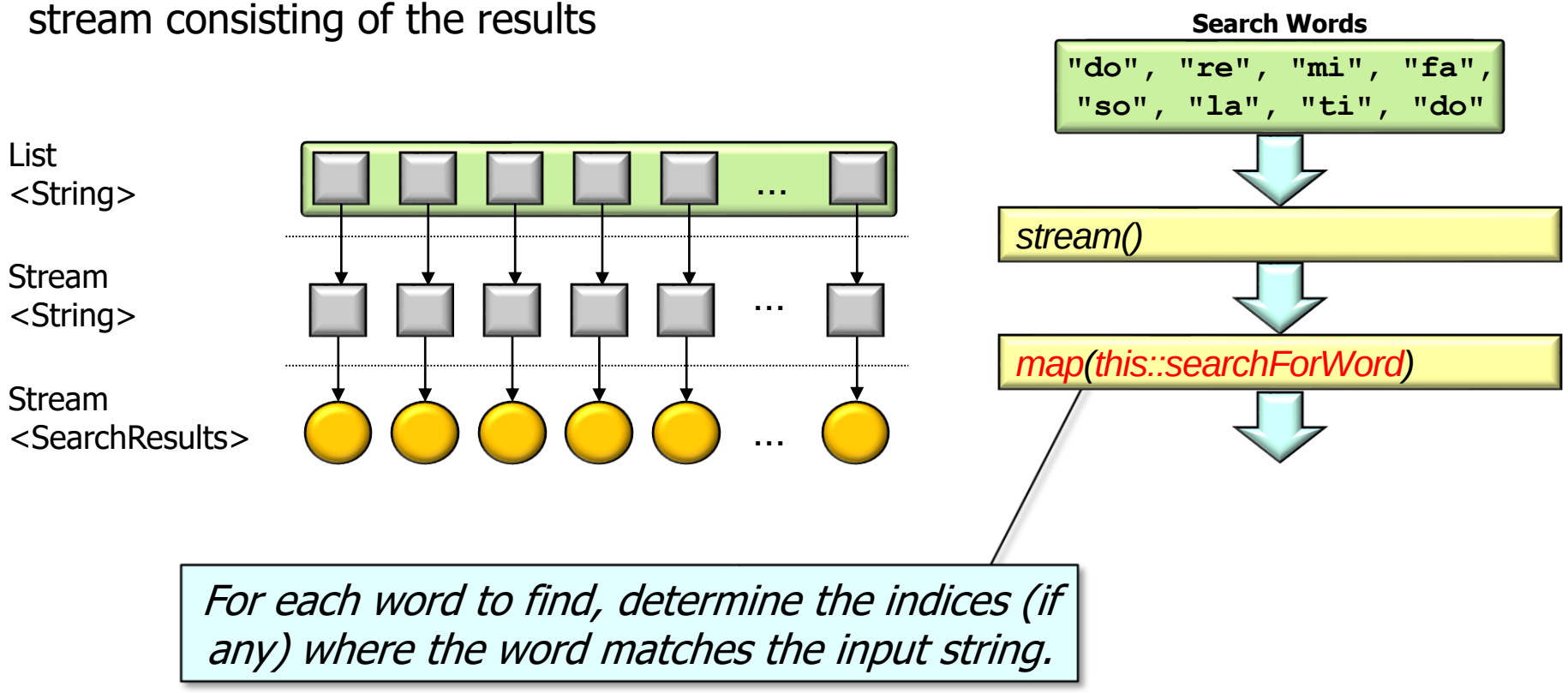




Naturally, a mapper may throw an exception, which could terminate map()

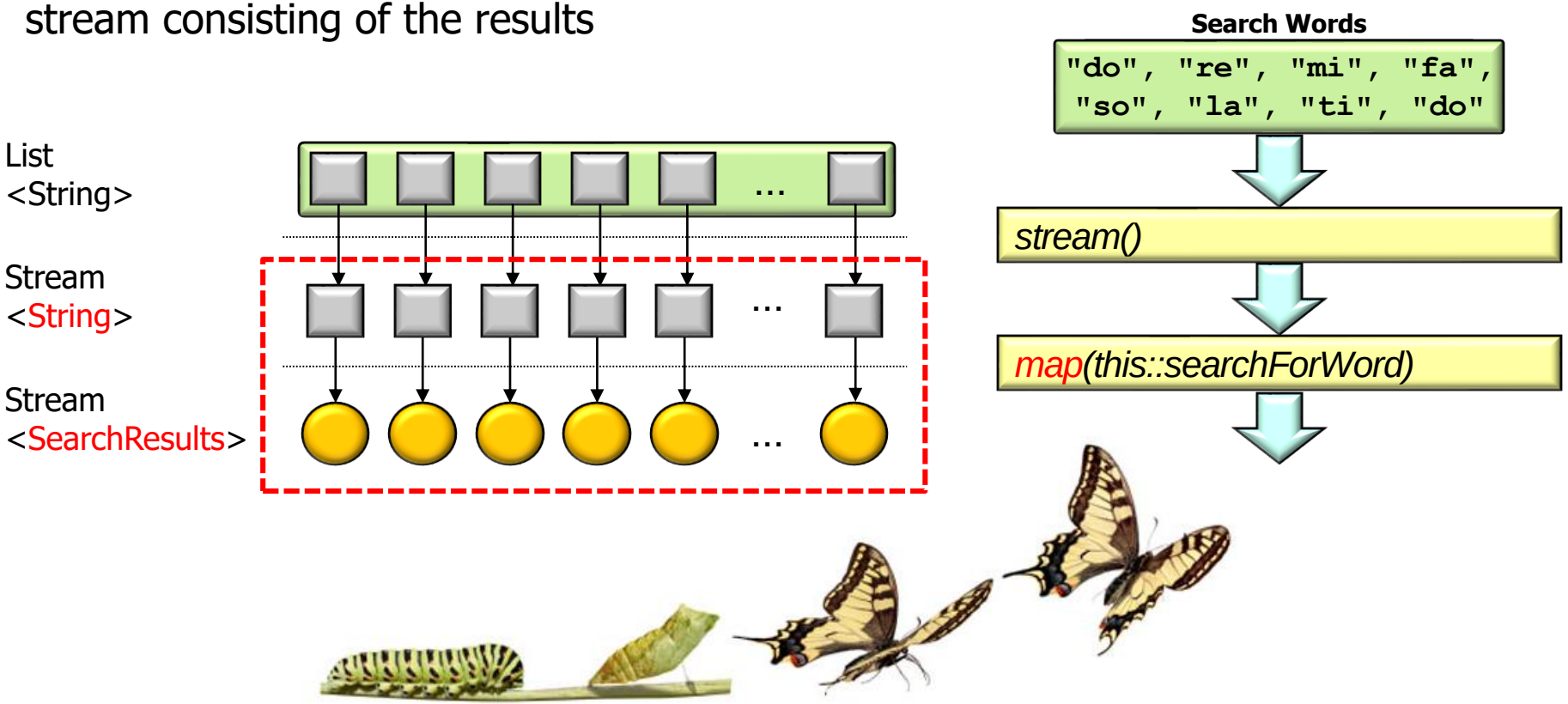
Overview of the map() Intermediate Operation

- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results



Overview of the map() Intermediate Operation

- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results

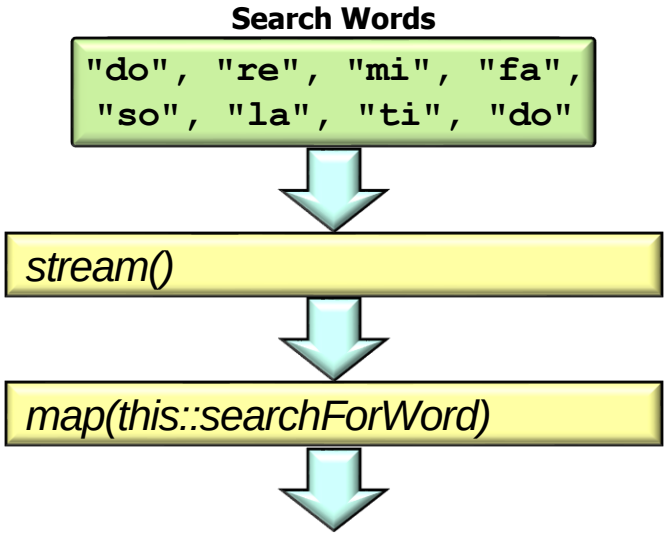


map() *may* transform the type of elements it processes

Overview of the map() Intermediate Operation

- Applies a mapper function to every element of the input stream & returns an output stream consisting of the results

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



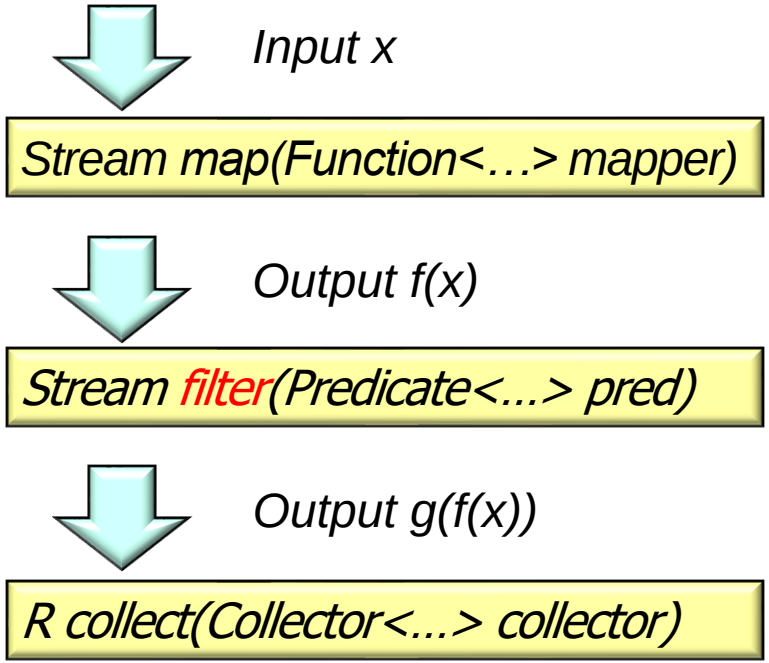
Note "fluent" programming style with cascading method calls.

See en.wikipedia.org/wiki/Fluent_interface

Overview of the filter() Intermediate Operation

Overview of the filter() Intermediate Operation

- Tests a predicate against each element of input stream & returns an output stream containing only elements that match the predicate

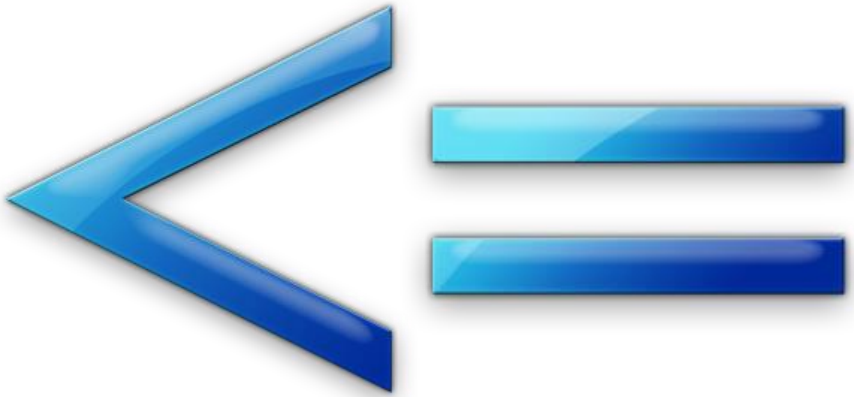


See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#filter

Overview of the filter() Intermediate Operation

- Tests a predicate against each element of input stream & returns an output stream containing only elements that match the predicate

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



Input x

Stream map(Function<…> mapper)

Output f(x)

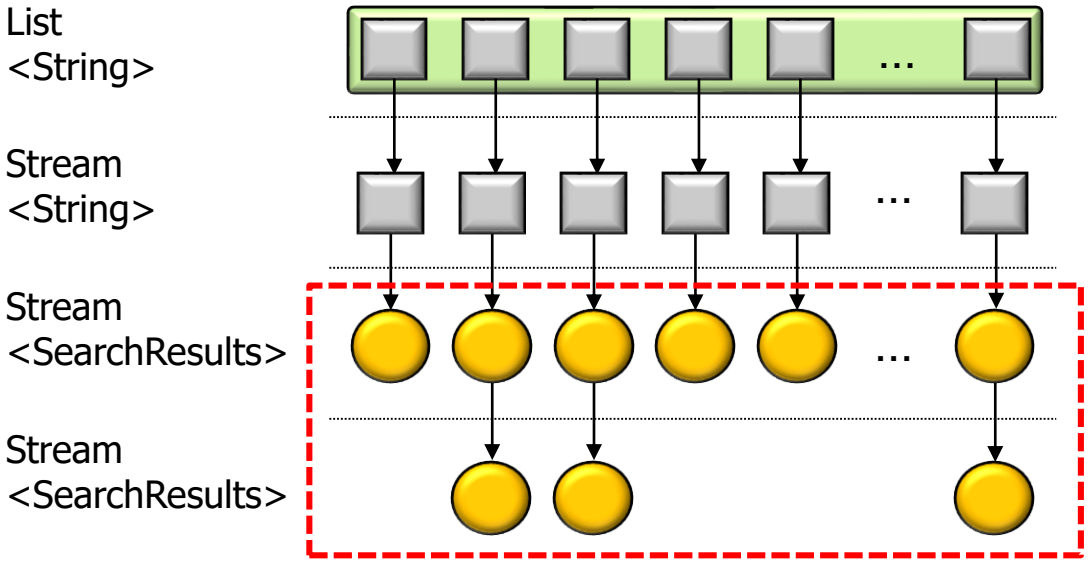
Stream filter(Predicate<…> pred)

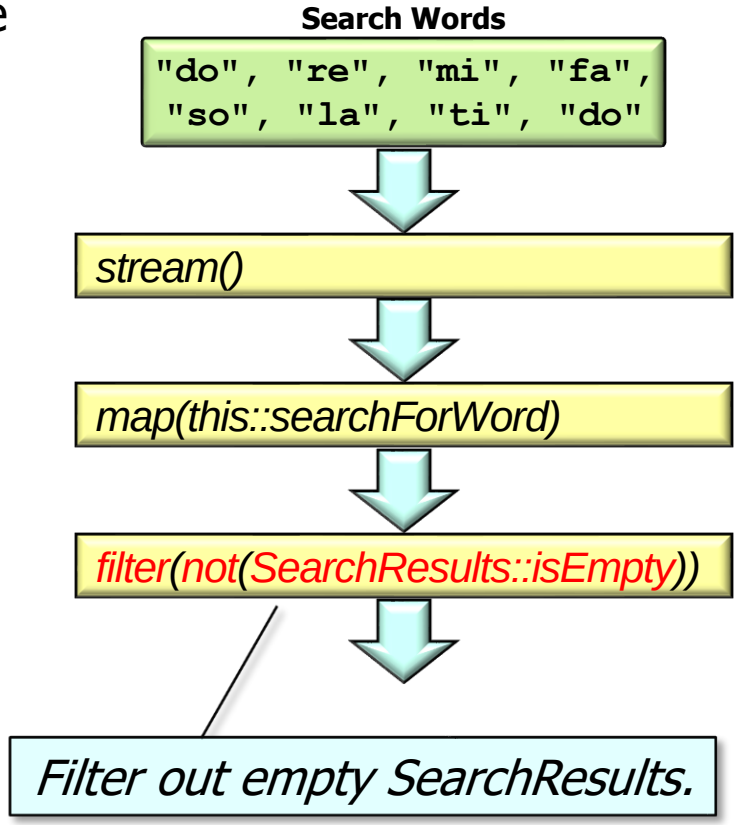
Output g(f(x))

R collect(Collector<…> collector)

Overview of the filter() Intermediate Operation

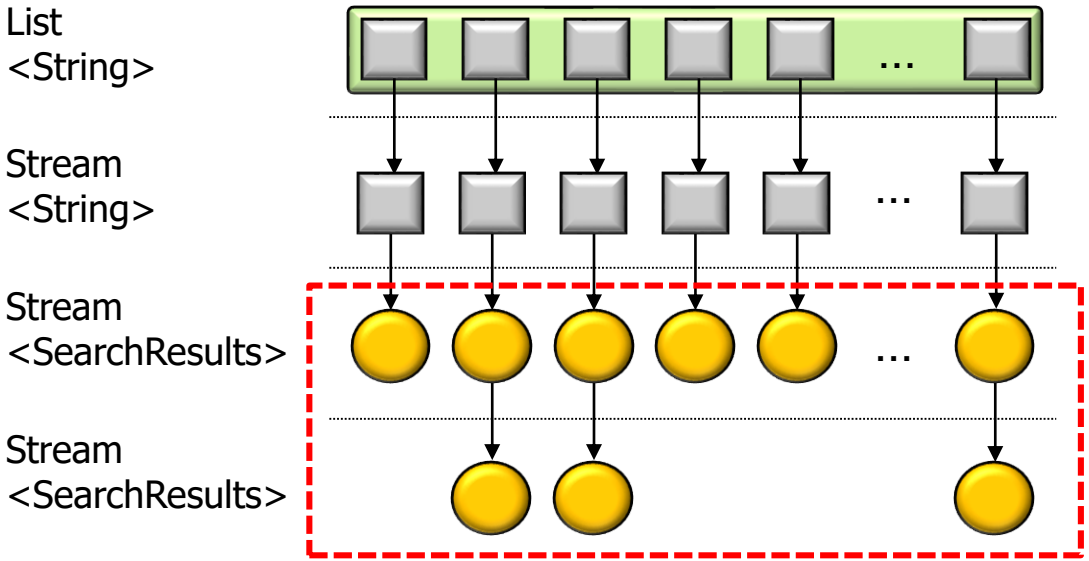
- Tests a predicate against each element of input stream & returns an output stream containing only elements that match the predicate

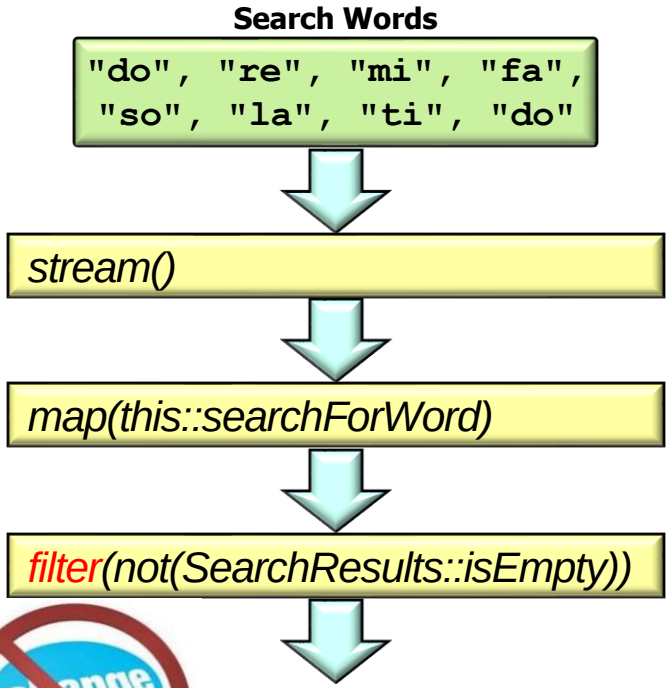




Overview of the filter() Intermediate Operation

- Tests a predicate against each element of input stream & returns an output stream containing only elements that match the predicate



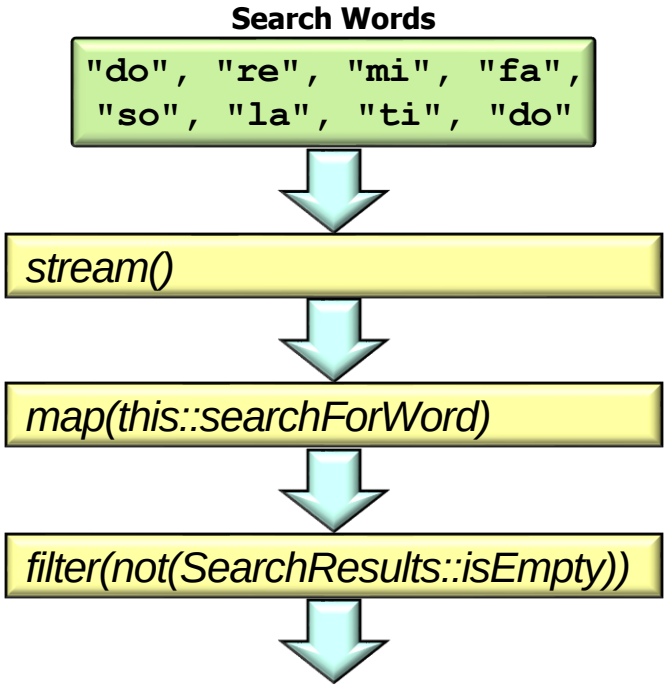


filter() *can't* change the type or value of elements it processes

Overview of the filter() Intermediate Operation

- Tests a predicate against each element of input stream & returns an output stream containing only elements that match the predicate

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



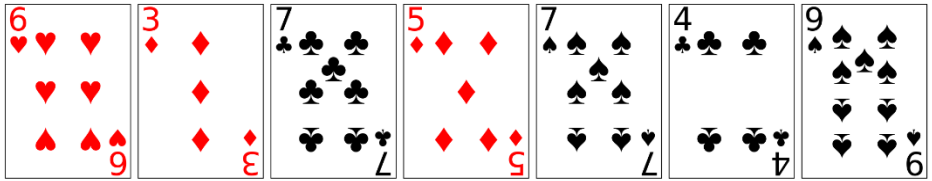
Again, note the fluent interface style.

See en.wikipedia.org/wiki/Fluent_interface

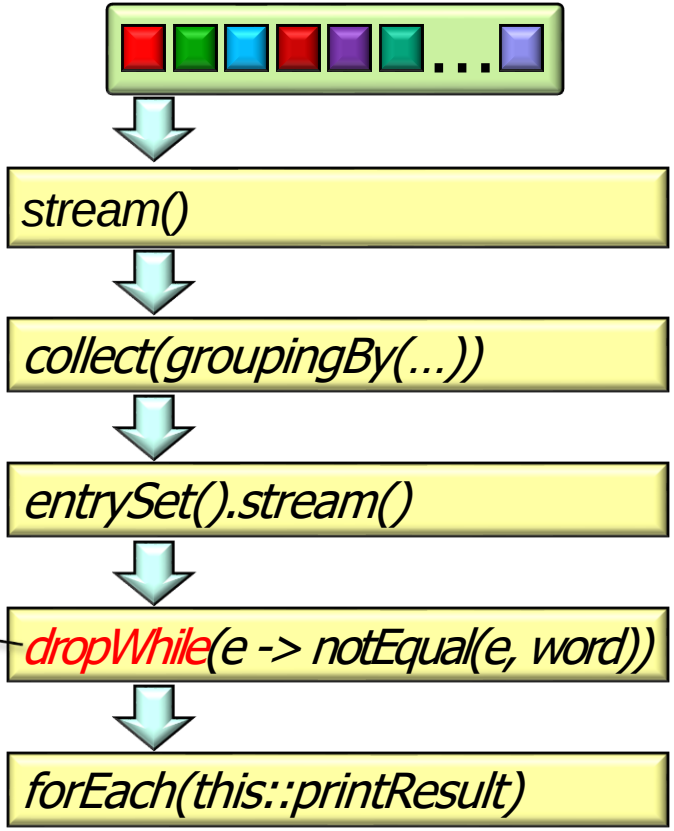
Overview of the dropWhile() Intermediate Operation

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)



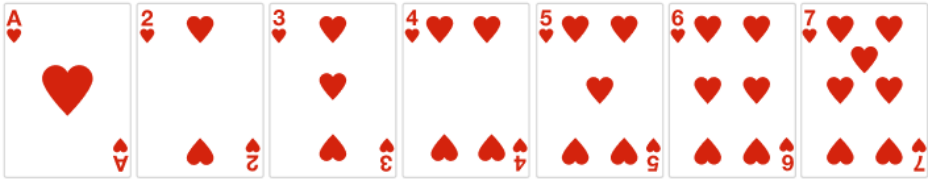
If this stream is unordered, return a stream consisting of the remaining elements of this stream after dropping a subset of elements that match the given predicate.



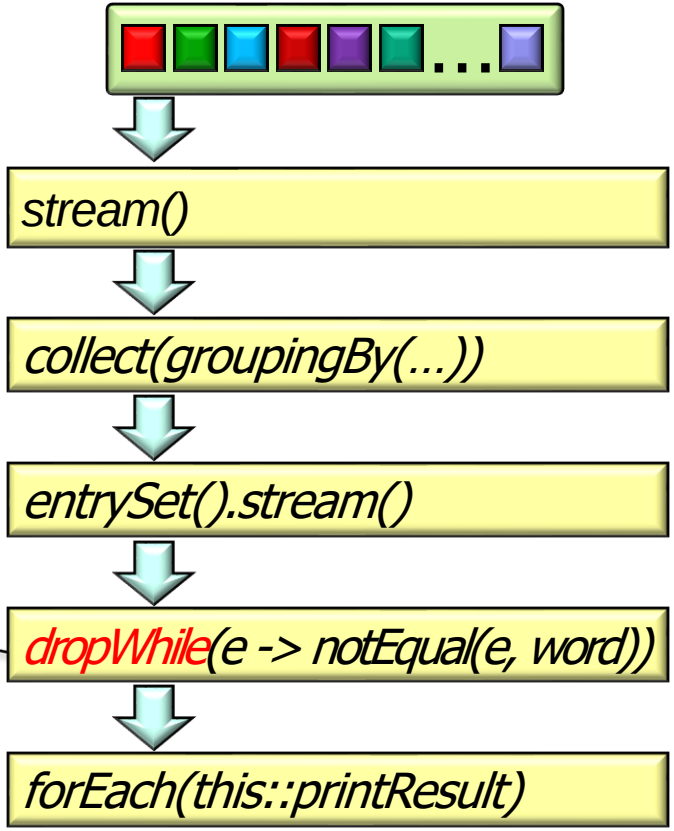
See docs.oracle.com/javase/9/docs/api/java/util/stream/Stream.html#dropWhile

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)



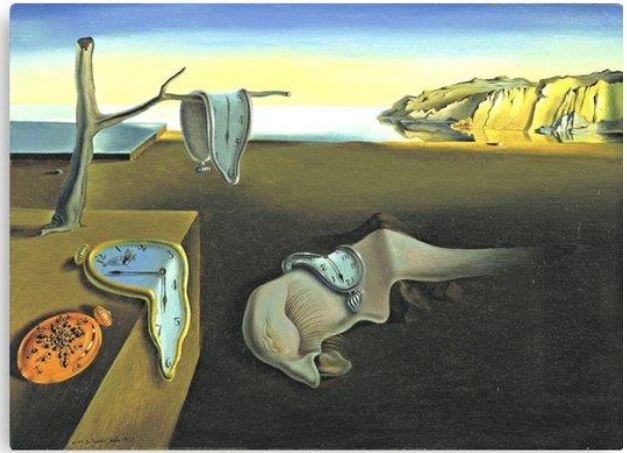
If this stream is ordered, return a stream consisting of the remaining elements of this stream after dropping the longest prefix of elements that match the given predicate.

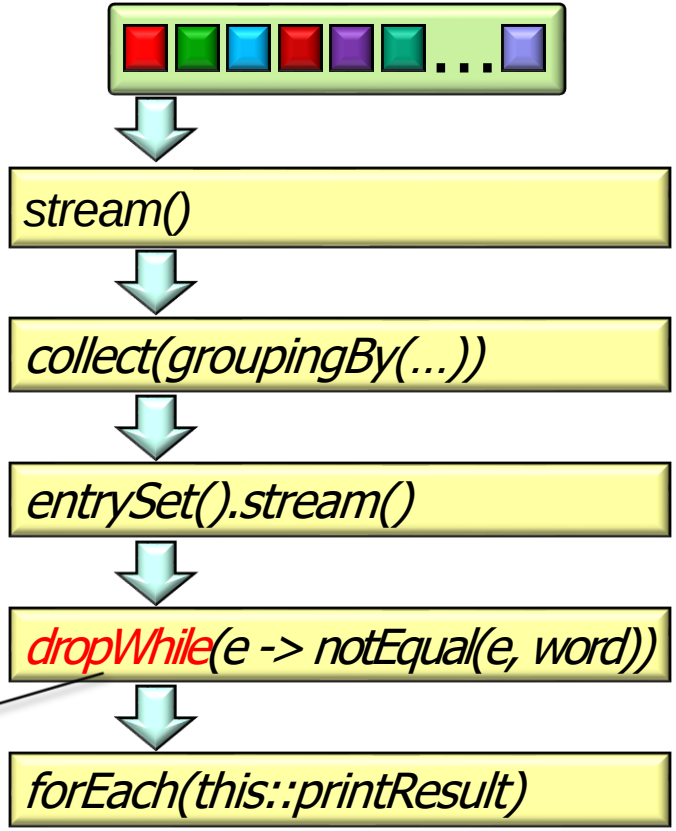


See docs.oracle.com/javase/9/docs/api/java/util/stream/Stream.html#dropWhile

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)



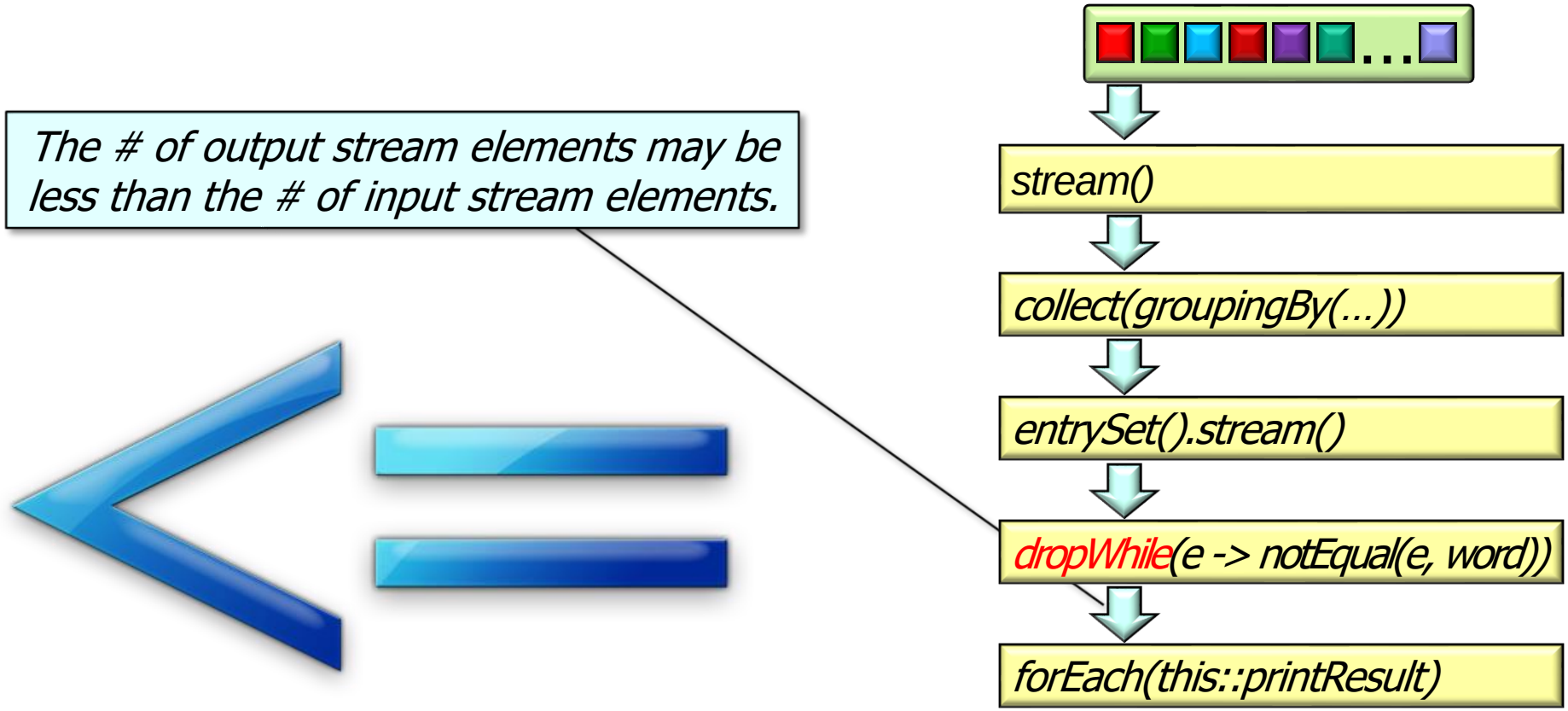


dropWhile() is a "stateful" operation that is costly on ordered parallel streams since threads must cooperate to find the longest contiguous sequence of matching elements in encounter order.

See blog.indrek.io/articles/whats-new-in-java-9-streams

Overview of the dropWhile() Intermediate Operation

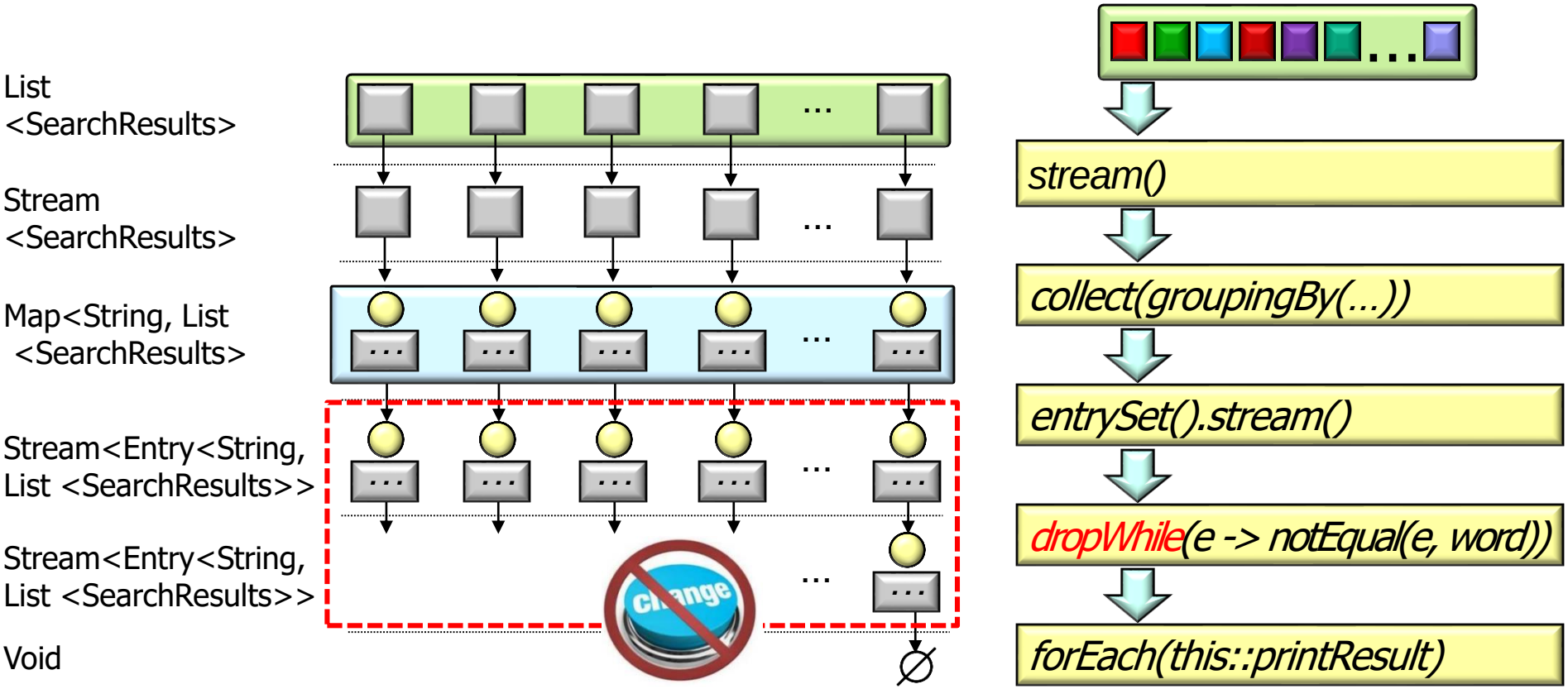
- Overview of the dropWhile() intermediate operation (introduced in Java 9)



However, the semantics of dropWhile() differ from the semantics of filter()..

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

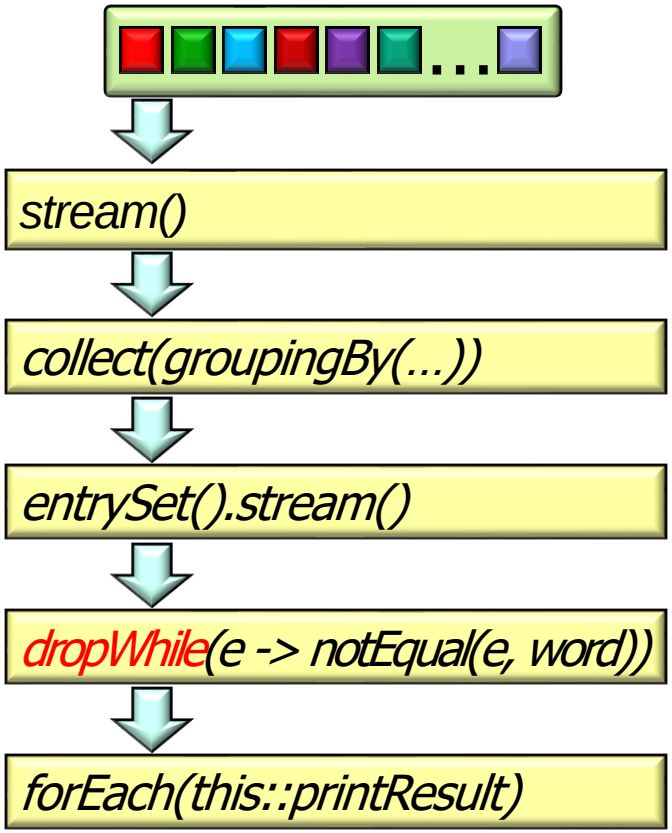


dropWhile() also *can't* change the type or values of elements it processes

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```

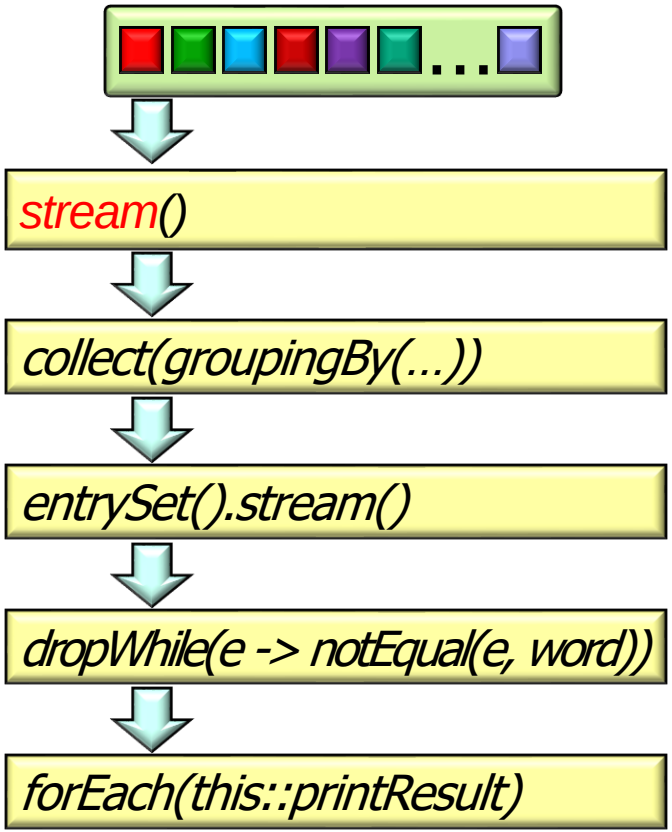


Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```

Convert list of search results into a stream

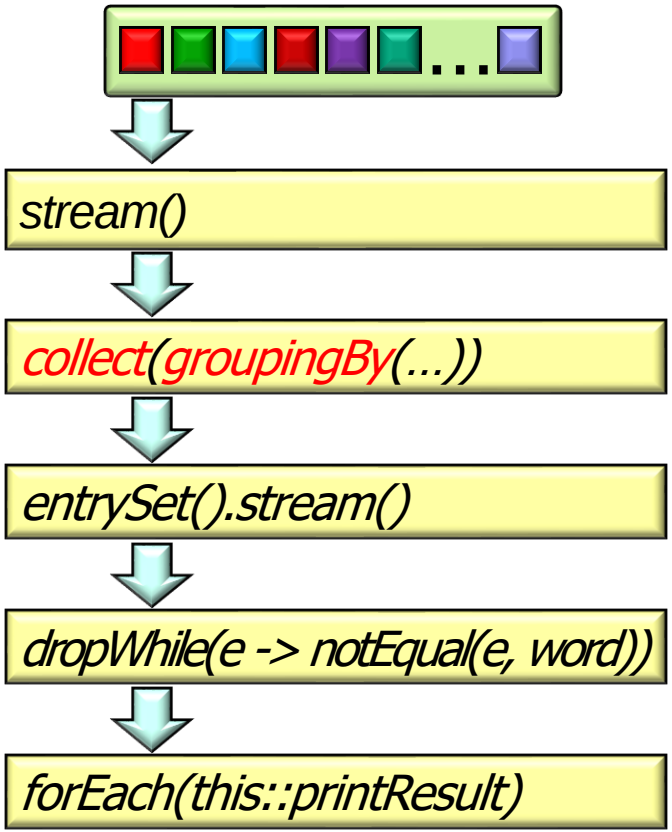


Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```

Collect stream into a map with words as key

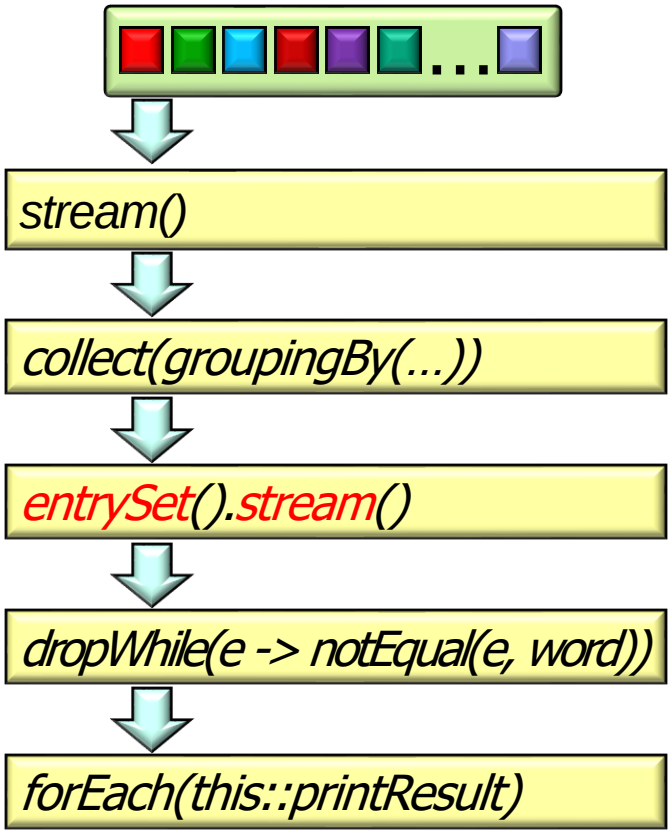


Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```

Convert map into a stream of entries

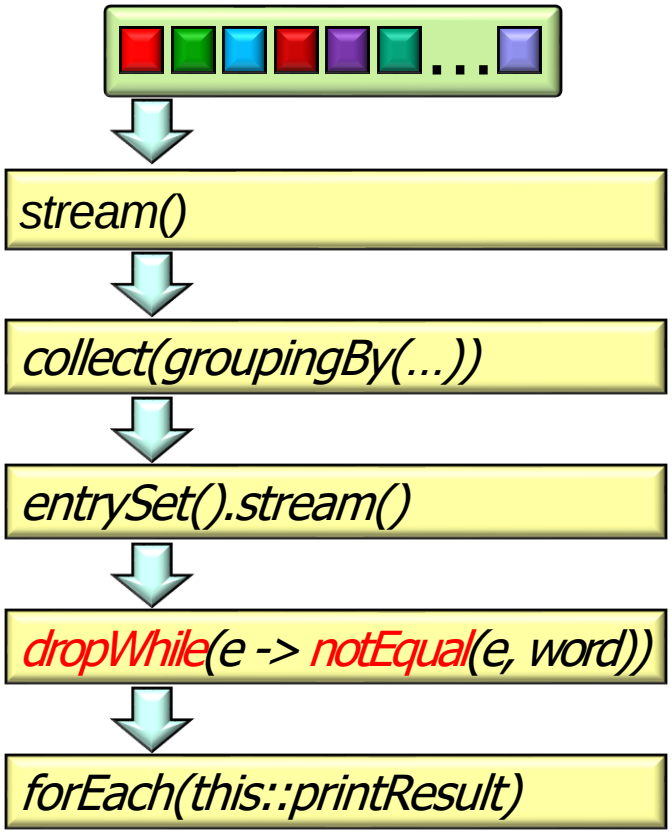


Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```

Ignore entries until there's a match

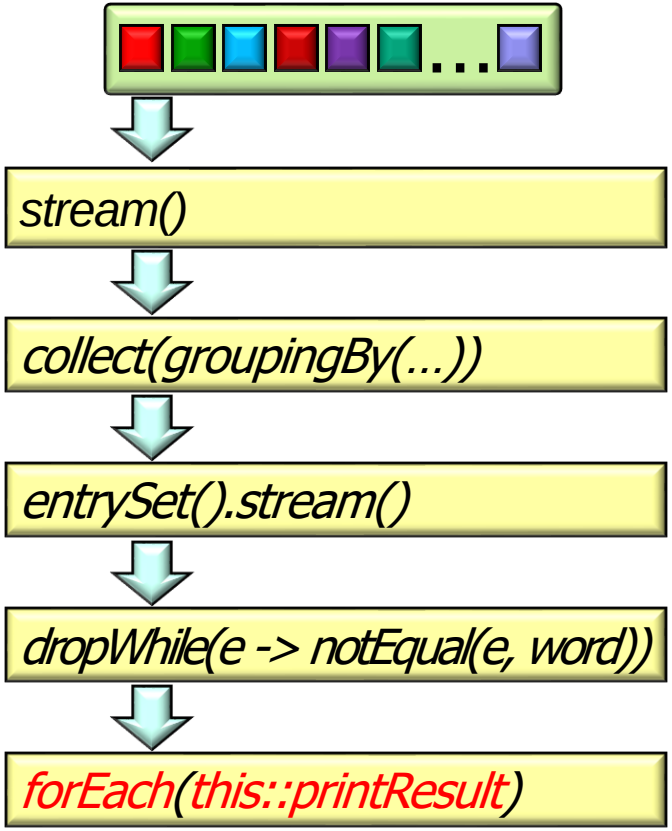


`notEqual()` is defined as `return !e.getKey().equals(word)`

Overview of the dropWhile() Intermediate Operation

- Overview of the dropWhile() intermediate operation (introduced in Java 9)

```
listOfResults
   .stream()
   .collect
      (groupingBy
         (SearchResults::getWord,
          LinkedHashMap::new,
          toList()))
   .entrySet()
   .stream()
   .dropWhile(e -> notEqual(e, word))
   .forEach(e -> printResult
                    (e.getKey(),
                     e.getValue()));
```



Print results starting at the match

End of Java Streams: Intermediate Operations