Java Streams Intermediate Operations map() & mapToInt()

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

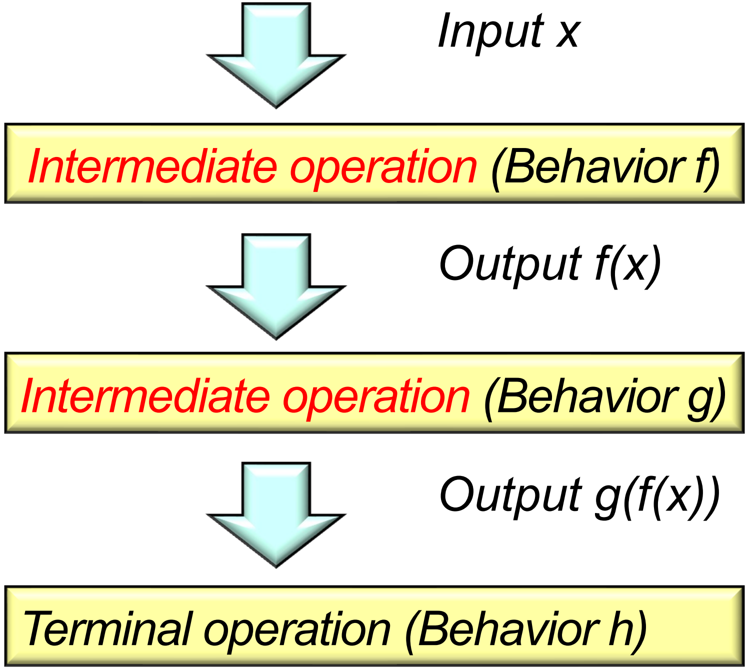
Professor of Computer Science

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



Input x

Intermediate operation (Behavior f)

Output f(x)

Intermediate operation (Behavior g)

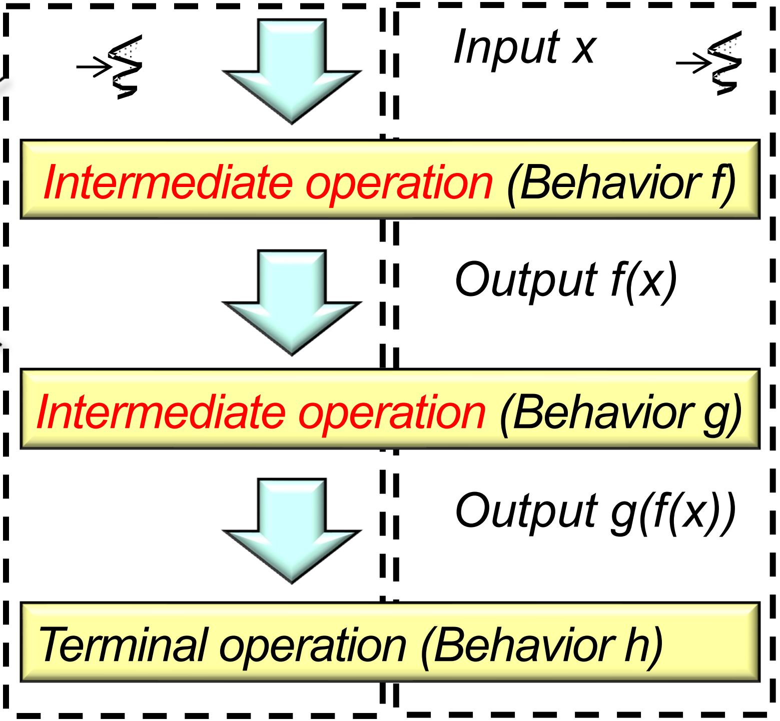
Output g(f(x))

Terminal operation (Behavior h)

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



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

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

Learning Objectives in this Part of the Lesson

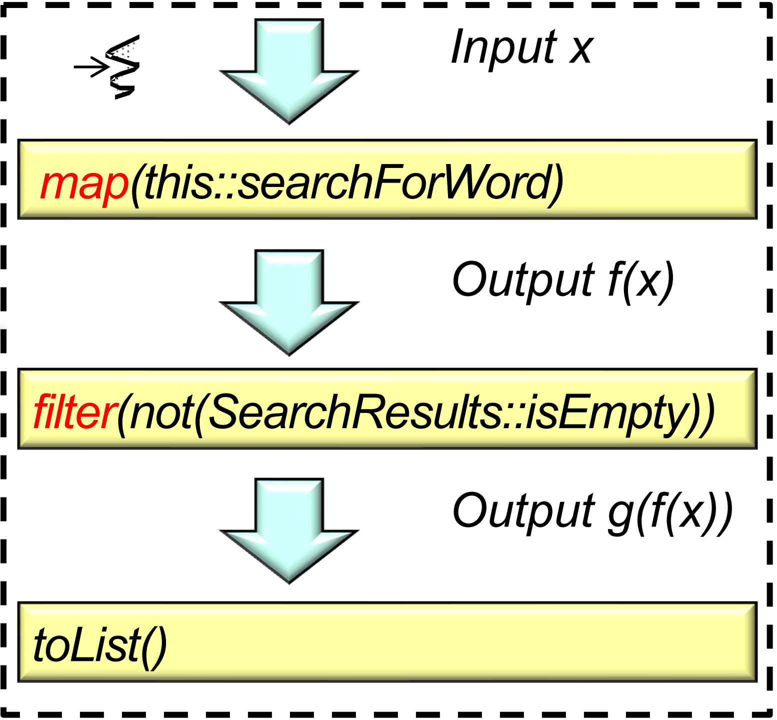
- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations

Input String to Search

Let's start at the very beginning..

Search Words

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

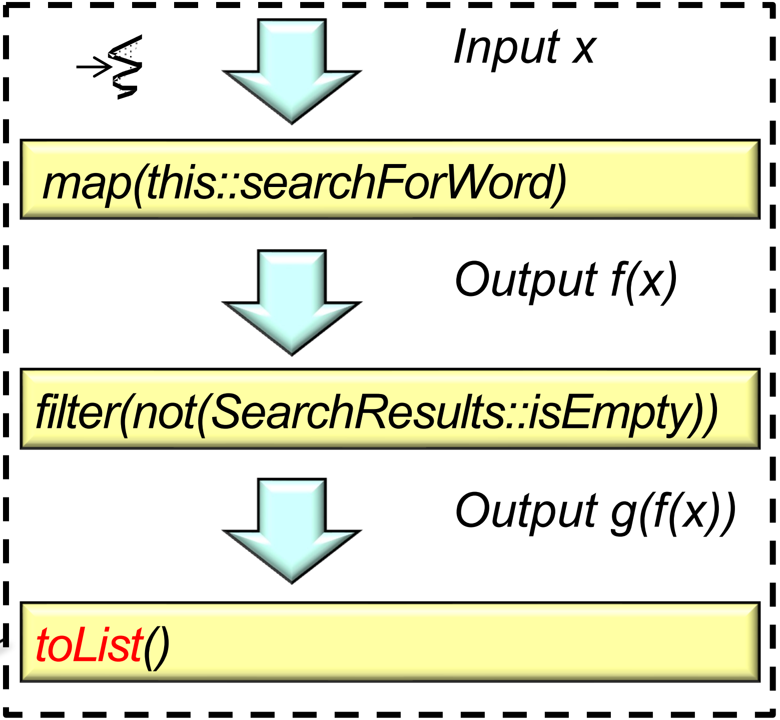


We continue to showcase the SimpleSearchStream program

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

Learning Objectives in this Part of the Lesson

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



Input x

map(this::searchForWord)

Output f(x)

filter(not(SearchResults::isEmpty))

Output g(f(x))

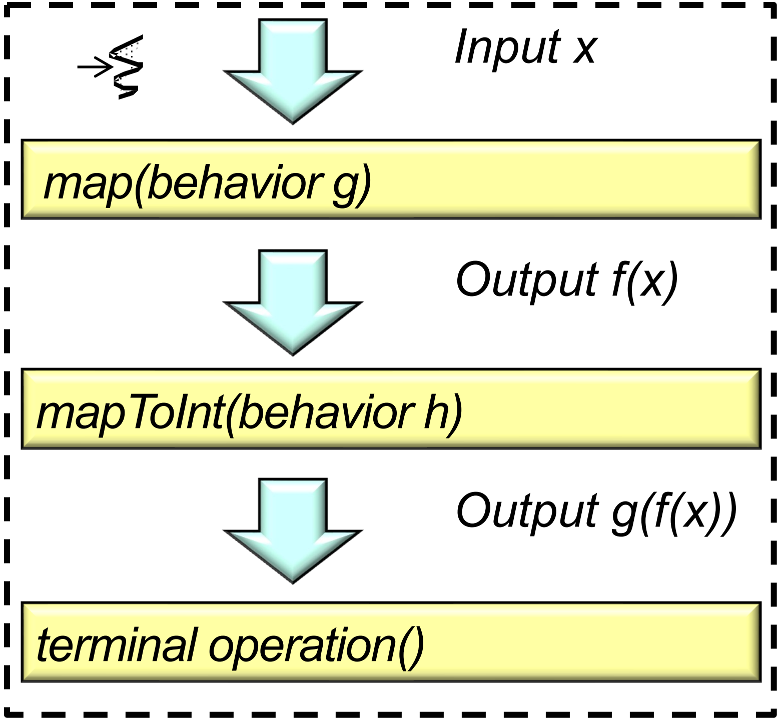
toList()

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

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

Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of stream aggregate operations
  - Intermediate operations
    - map() & mapToInt()

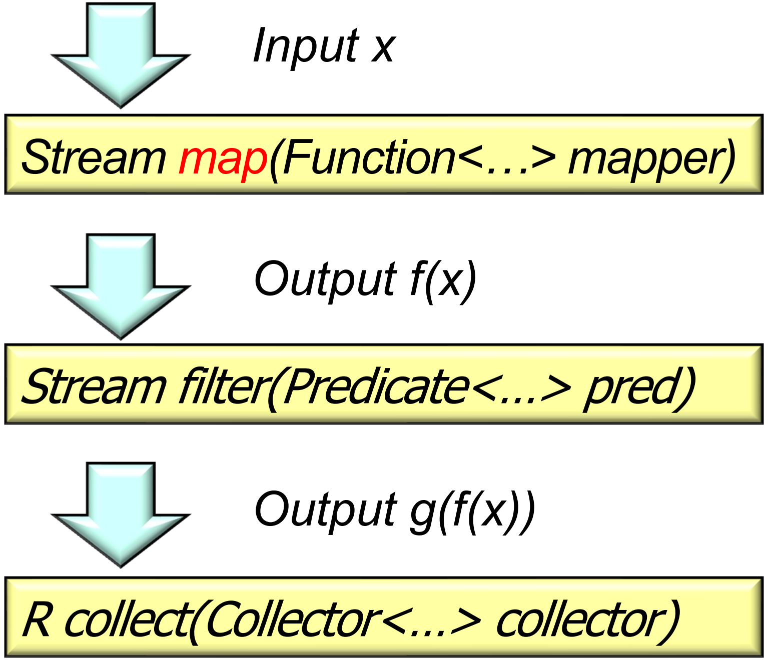


These are both stateless, run-to-completion operations

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



Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<…> pred)

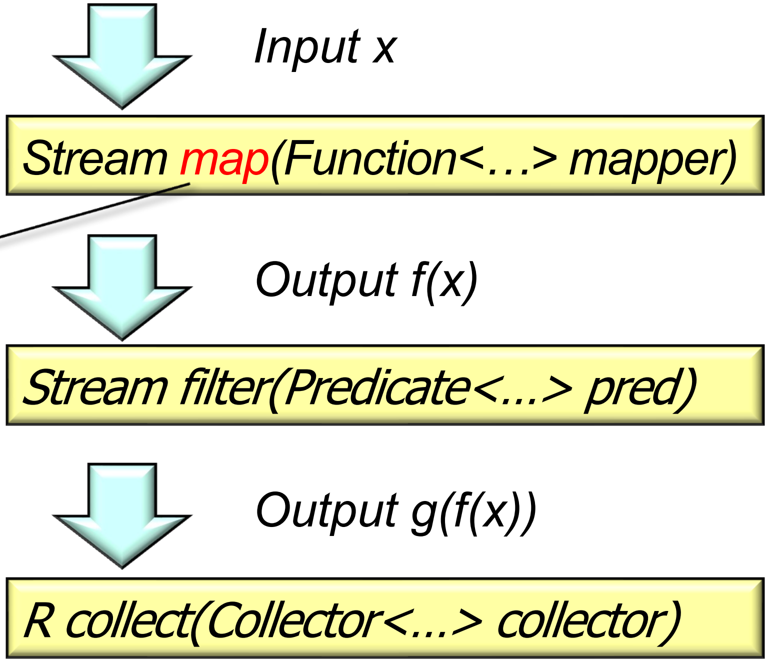
Output g(f(x))

R collect(Collector<…> collector)

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



Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<...> pred)

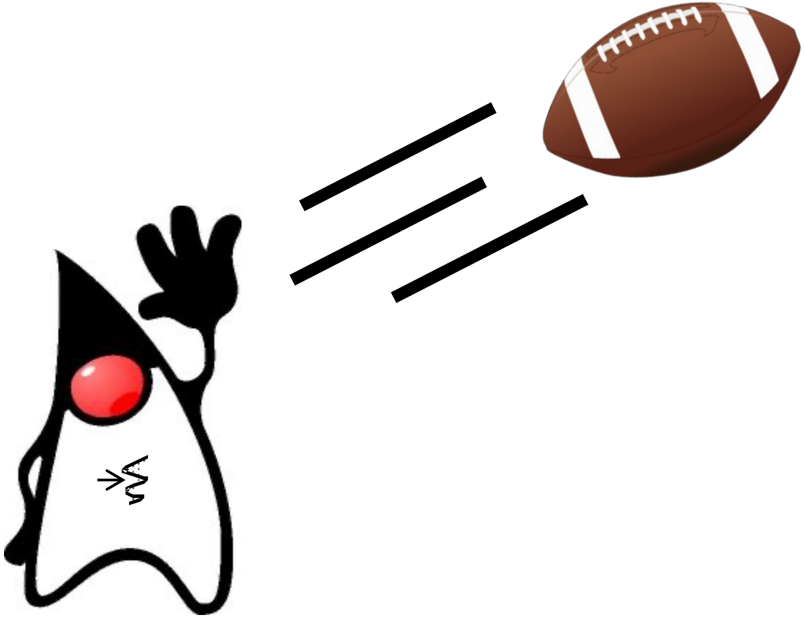
Output g(f(x))

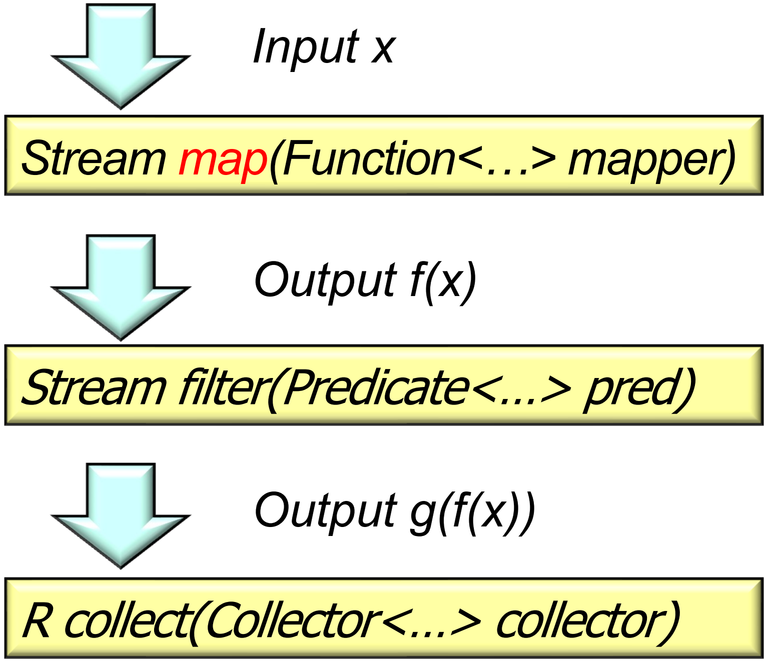
R collect(Collector<...> collector)

The # of output stream elements must match the # of input stream elements.

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
  - A mapper may throw an exception, which could terminate map()





Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<…> pred)

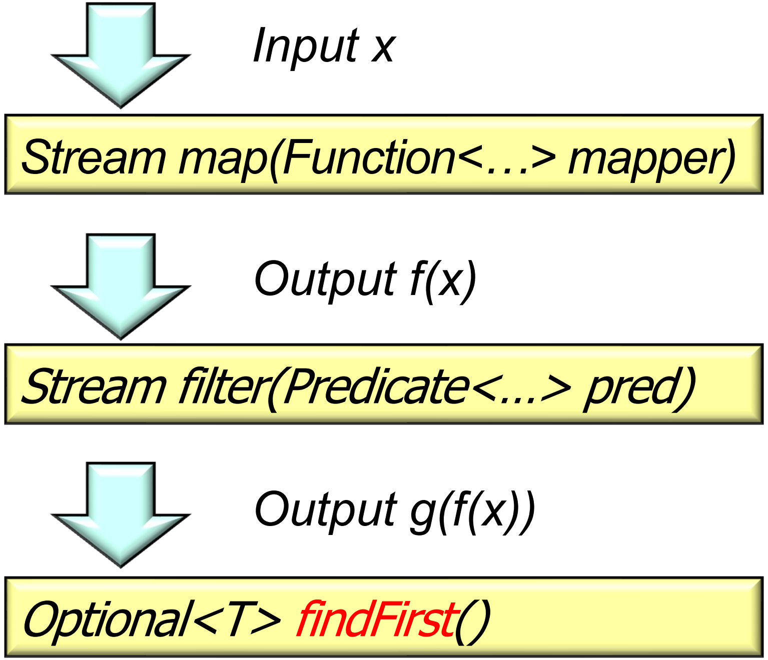
Output g(f(x))

R collect(Collector<…> collector)

See dzone.com/articles/exception-handling-in-java-streams

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
  - A mapper may throw an exception, which could terminate map()
  - A short-circuit terminal operation also causes the map() operation to only process a subset of the input stream

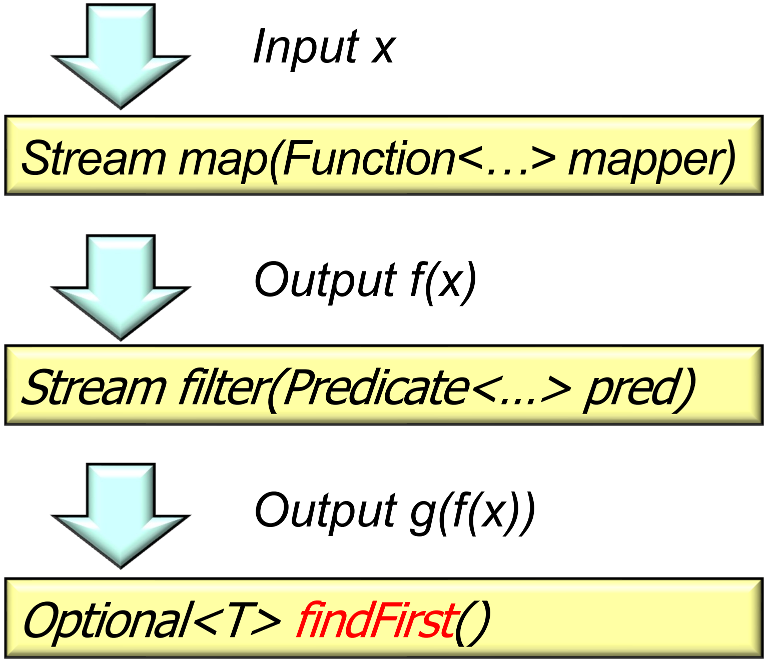


Input x
Stream map(Function<…> mapper)
Output f(x)
Stream filter(Predicate<…> pred)
Output g(f(x))
Optional<T> findFirst()

See dzone.com/articles/collectors-part-1-%E2%80%93-reductions

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
  - A mapper may throw an exception, which could terminate map()
  - A short-circuit terminal operation also causes the map() operation to only process a subset of the input stream


Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<…> pred)

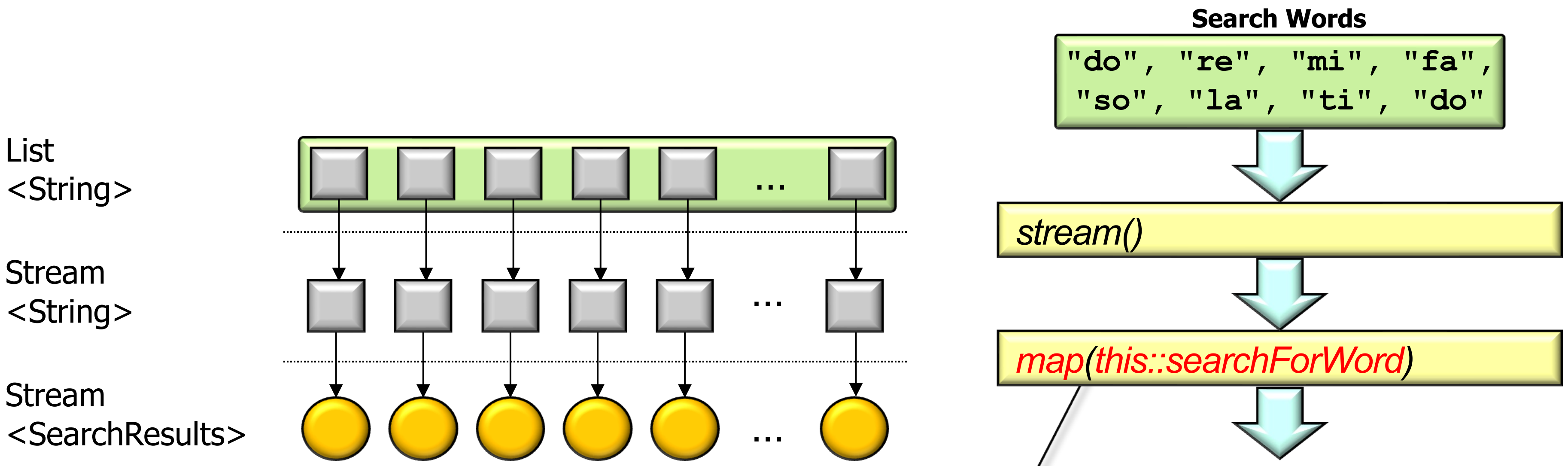
Output g(f(x))

Optional<T> findFirst()

These caveats apply to all "run-to-completion" intermediate operations!

Overview of the map() Intermediate Operation

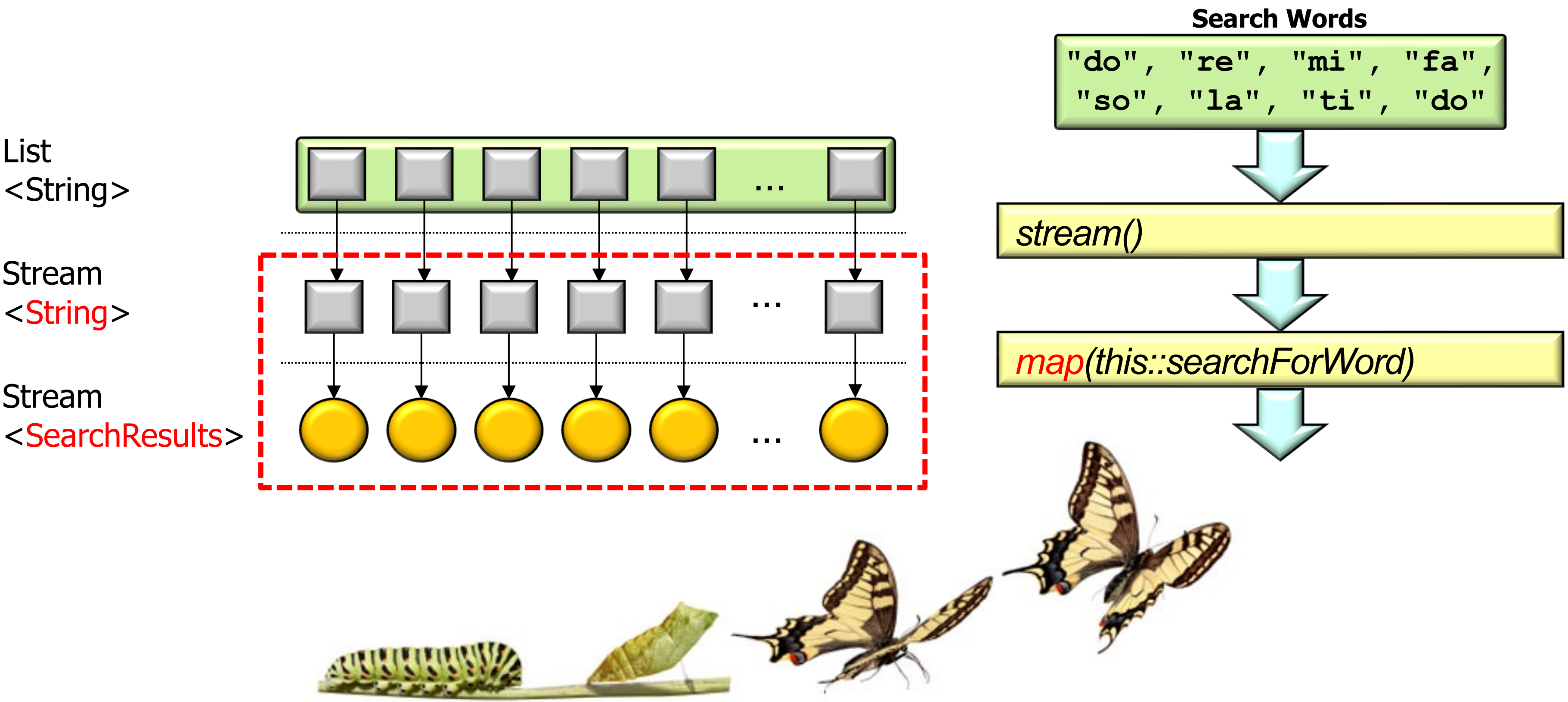
- Example of applying map() & a mapper function in the SimpleSearchStream program



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

Overview of the map() Intermediate Operation

- Example of applying map() & a mapper function in the SimpleSearchStream program

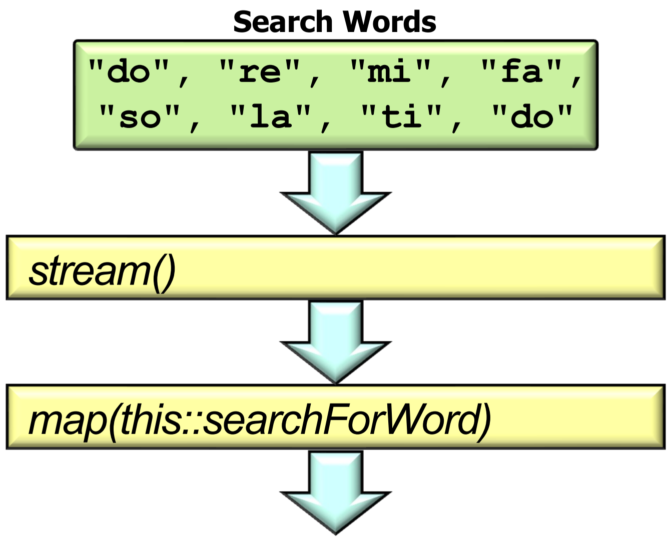


map() *may* transform the type of elements it processes

Overview of the map() Intermediate Operation

- Example of applying map() & a mapper function in the SimpleSearchStream program

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



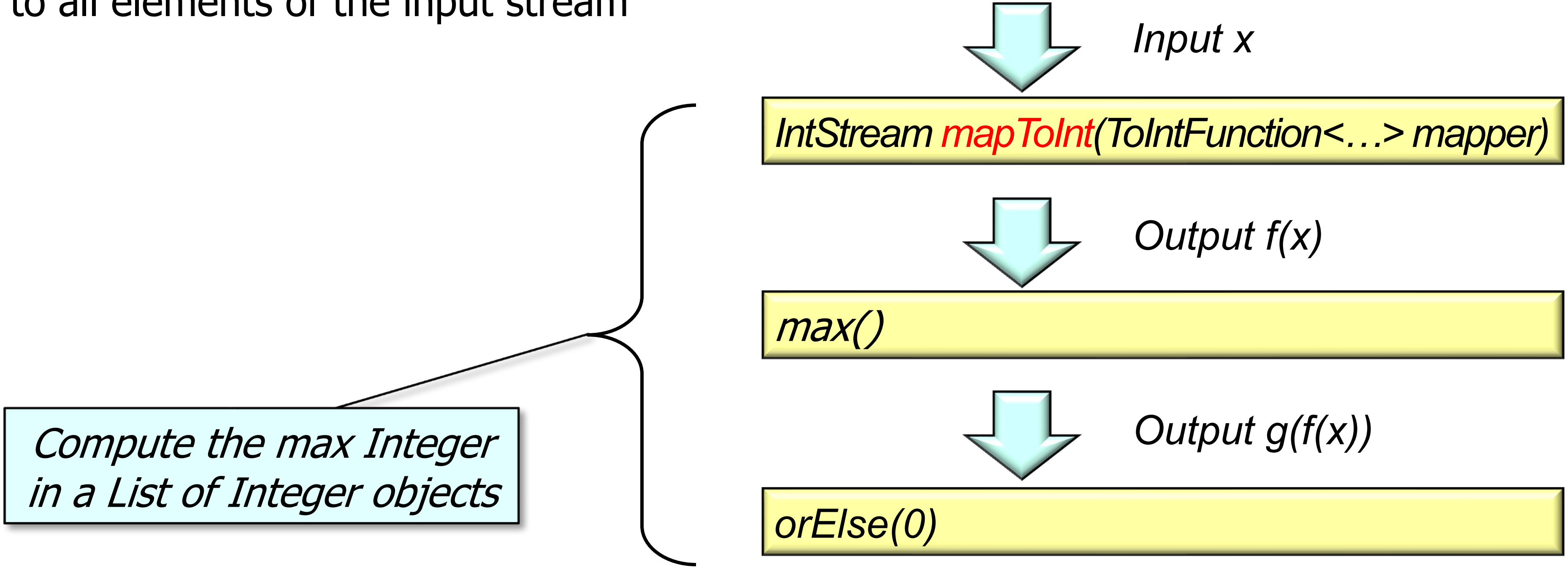
Note "fluent" programming style with cascading method calls.

See en.wikipedia.org/wiki/Fluent_interface

Overview of the mapToInt() Intermediate Operation

Overview of the mapToInt() Intermediate Operation

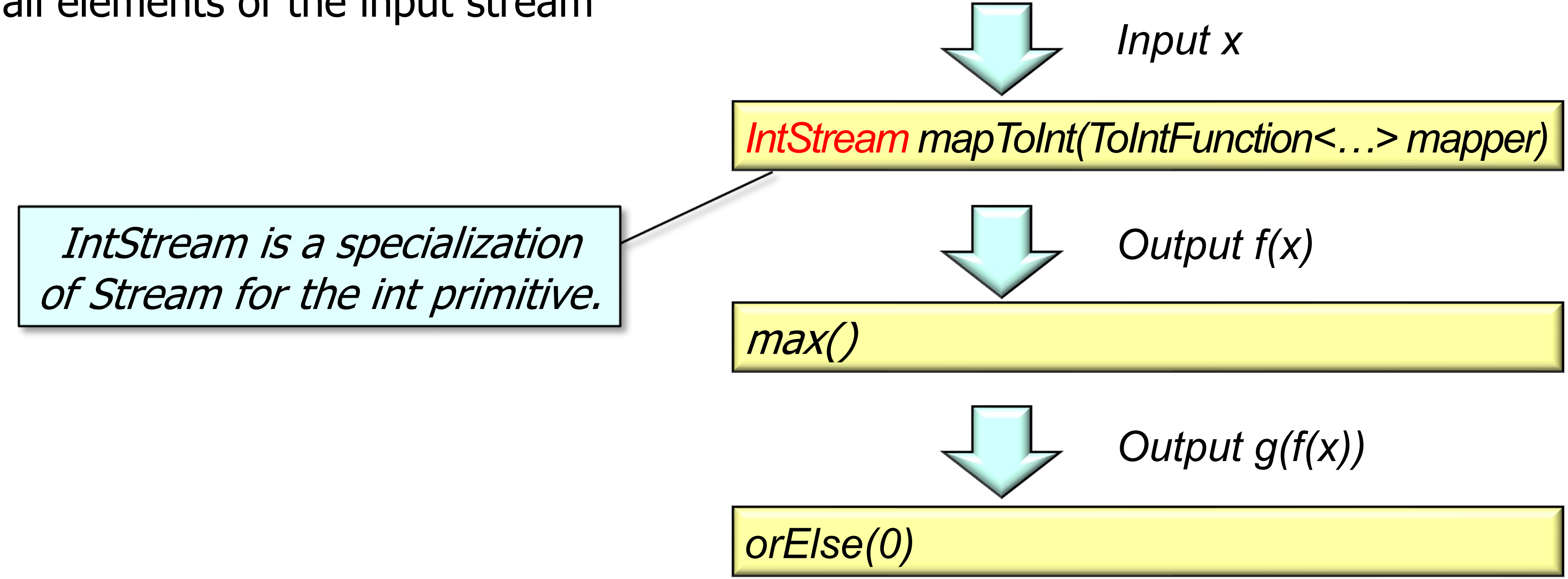
- Returns an IntStream consisting of the results of applying the given mapper function to all elements of the input stream


Input x
IntStream mapToInt(ToIntFunction<…> mapper)
Output f(x)
max()
Output g(f(x))
orElse(0)
Compute the max Integer in a List of Integer objects

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

Overview of the mapToInt() Intermediate Operation

- Returns an IntStream consisting of the results of applying the given mapper function to all elements of the input stream



Input x

IntStream mapToInt(ToIntFunction<…> mapper)

IntStream is a specialization of Stream for the int primitive.

Output f(x)

max()

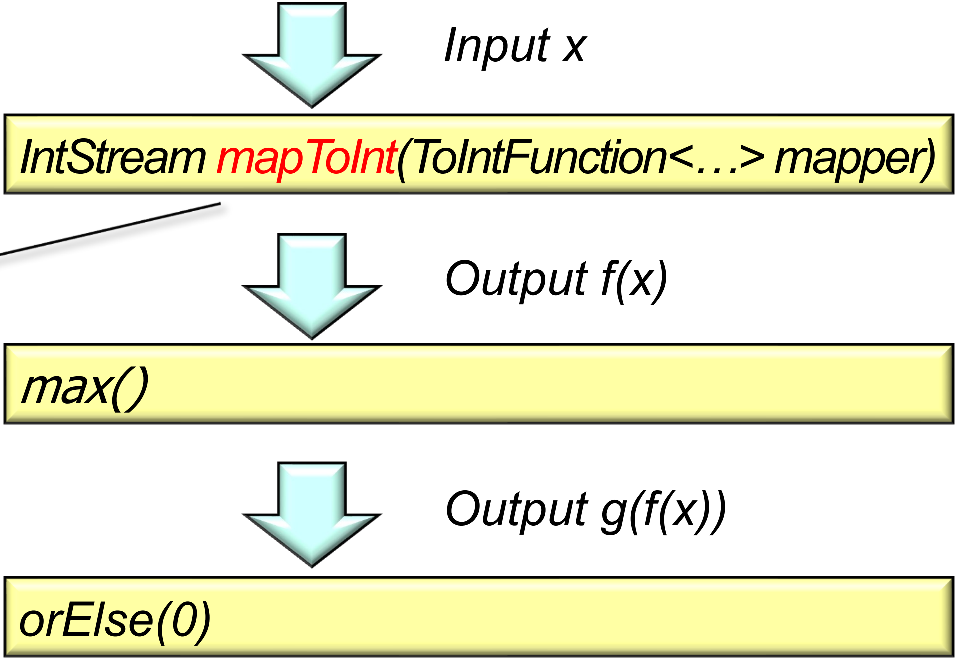
Output g(f(x))

orElse(0)

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

Overview of the mapToInt() Intermediate Operation

- Returns an IntStream consisting of the results of applying the given mapper function to all elements of the input stream



Input x

IntStream mapToInt(ToIntFunction<…> mapper)

Output f(x)

max()

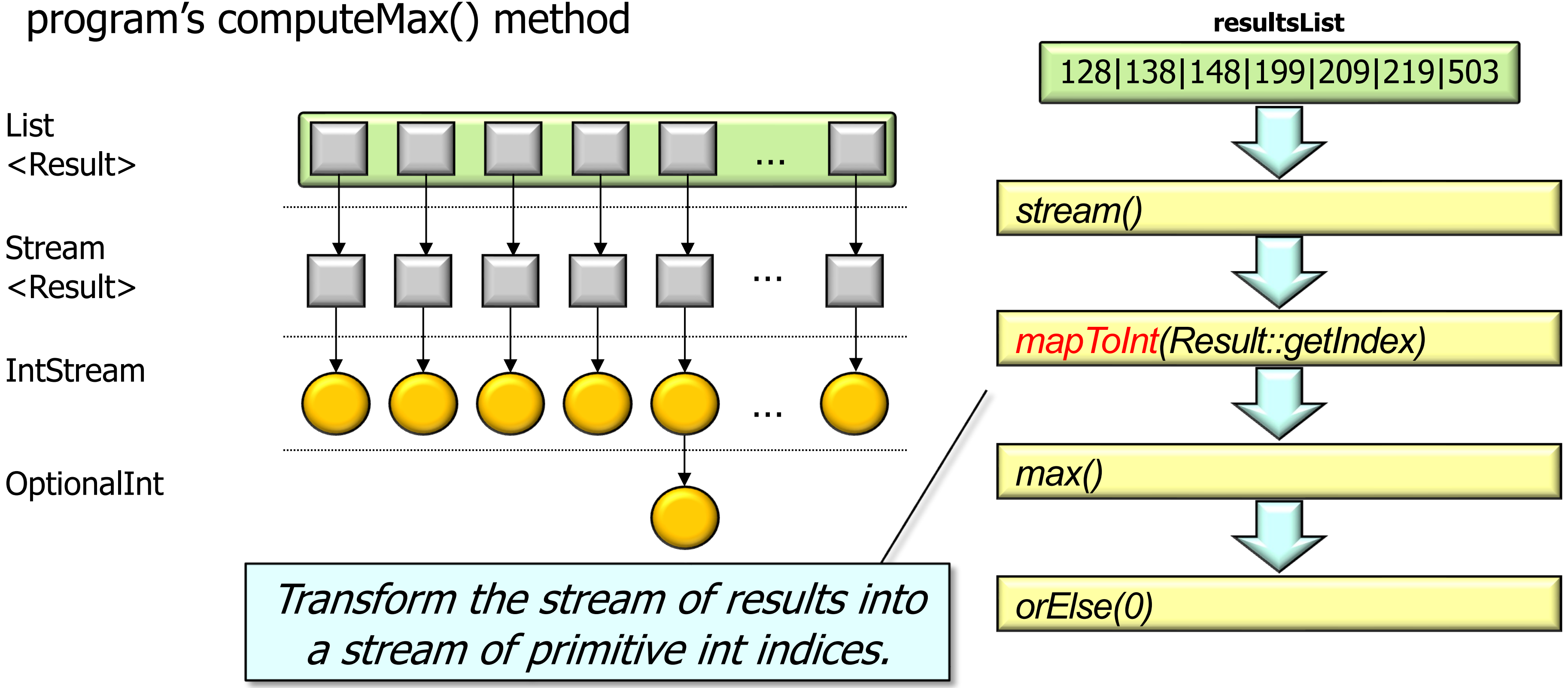
Output g(f(x))

orElse(0)

The # of output stream elements must match the # of input stream elements.

Overview of the mapToInt() Intermediate Operation

- Example of applying mapToInt() & a mapper function in the SimpleSearchStream program's computeMax() method



List <Result>

Stream <Result>

IntStream

OptionalInt

resultsList

128|138|148|199|209|219|503

stream()

mapToInt(Result::getIndex)

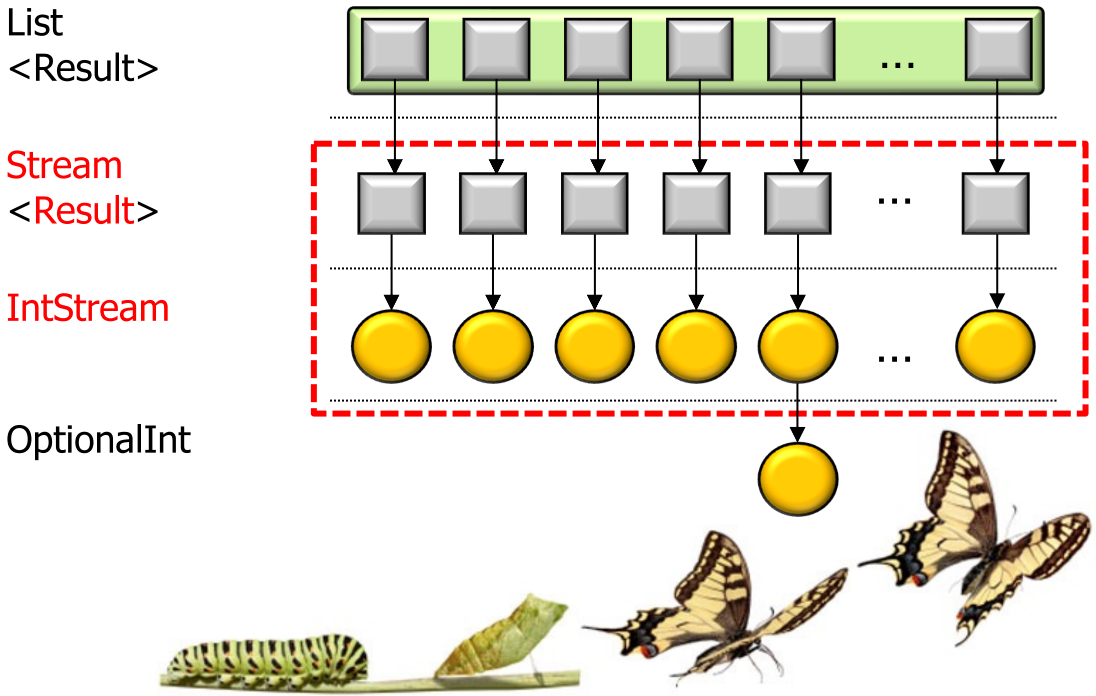
max()

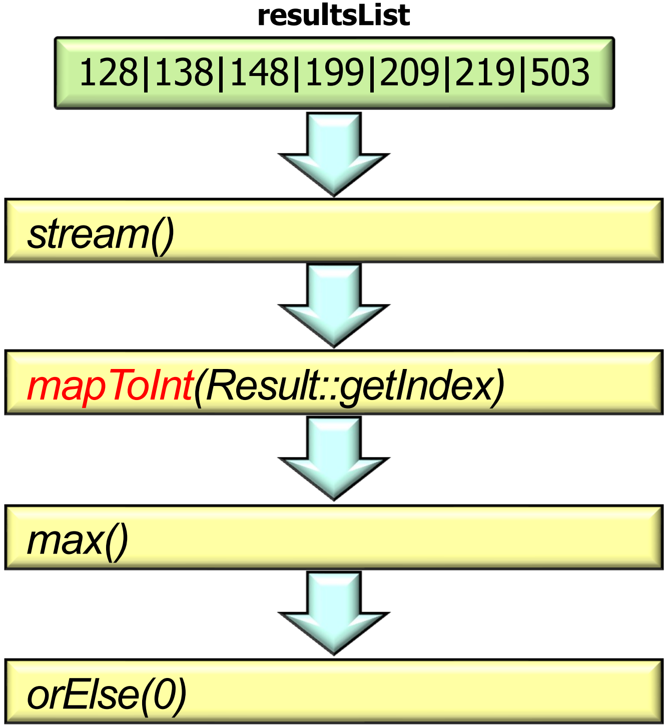
orElse(0)

Transform the stream of results into a stream of primitive int indices.

Overview of the mapToInt() Intermediate Operation

- Example of applying mapToInt() & a mapper function in the SimpleSearchStream program's computeMax() method




resultsList

128|138|148|199|209|219|503

stream()

mapToInt(Result::getIndex)

max()

orElse(0)

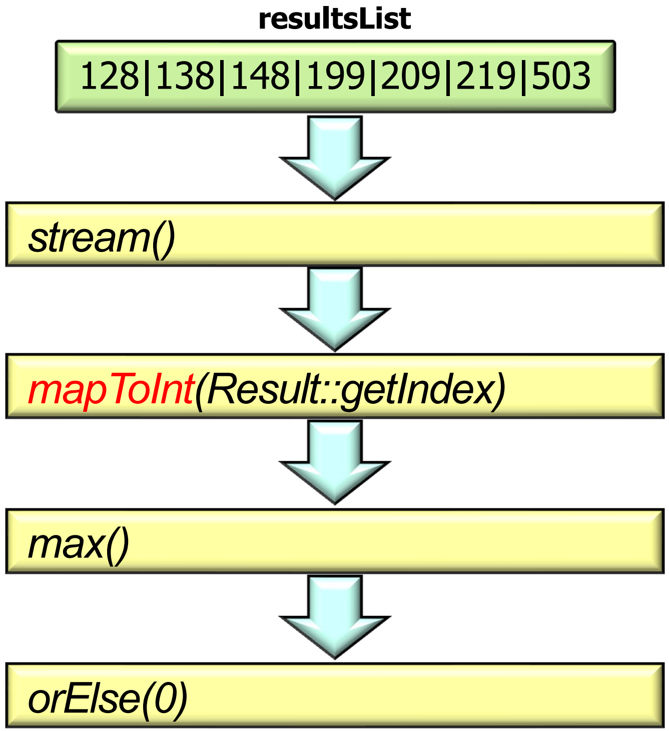
mapToInt() transforms the type of elements it processes into primitive ints

Overview of the mapToInt() Intermediate Operation

- Example of applying mapToInt() & a mapper function in the SimpleSearchStream program's computeMax() method

```
int computeMax
          (List<SearchResults.Result>
           resultsList) {
  return resultsList
    .stream()
    .mapToInt(SearchResults.Result
                       ::getIndex)
    .max()
    .orElse(0);
}
```

Note "fluent" programming style with cascading method calls.



resultsList

128|138|148|199|209|219|503

stream()

mapToInt(Result::getIndex)

max()

orElse(0)

See en.wikipedia.org/wiki/Fluent_interface

End of Java Streams Intermediate Operations map() & mapToInt()