Java Streams: Intermediate Operations (Part 1)

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

\[\text{Input } x\]

**Intermediate operation (Behavior } f)\]

\[\text{Output } f(x)\]

**Intermediate operation (Behavior } g)\]

\[\text{Output } g(f(x))\]

**Terminal operation (Behavior } h)\]

2
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.

Learning Objectives in this Part of the Lesson

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

\[ \text{Input } x \rightarrow \text{Stream } \text{map}(\text{Function}<\ldots> \text{mapper}) \rightarrow \text{Output } f(x) \rightarrow \text{Stream } \text{filter}(\text{Predicate}<\ldots> \text{pred}) \rightarrow \text{Output } g(f(x)) \rightarrow R \text{ collect}(\text{Collector}<\ldots> \text{collector}) \]

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

\[ \text{Input } x \rightarrow \text{Stream } \text{map}(\text{Function}<\ldots> \text{ mapper}) \rightarrow \text{Output } f(x) \rightarrow \text{Stream filter}(\text{Predicate}<\ldots> \text{ pred}) \rightarrow \text{Output } g(f(x)) \rightarrow R \text{ collect}(\text{Collector}<\ldots> \text{ collector}) \]

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

\[
\text{Input } x \quad \rightarrow \quad \text{Stream } \text{map}(\text{Function}<\ldots> \text{mapper})
\]

\[
\text{Output } f(x) \quad \rightarrow \quad \text{Stream } \text{filter}(\text{Predicate}<\ldots> \text{pred})
\]

\[
\text{Output } g(f(x)) \quad \rightarrow \quad R \ \text{collect}(\text{Collector}<\ldots> \text{collector})
\]

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

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

- List & Stream with String types are transformed into a Stream with SearchResults type using the map() function.

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

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

See [en.wikipedia.org/wiki/Fluent_interface](en.wikipedia.org/wiki/Fluent_interface)

Note “fluent” programming style with cascading method calls.
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(TToIntFunction<...> mapper)
Output f(x)
max()
Output g(f(x))
orElse(0)
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#mapToInt](https://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

\[
\text{Input } x \quad \Rightarrow \quad \text{IntStream } \text{mapToInt}(\text{ToIntFunction}<\ldots> \text{ mapper})
\]

\[
\text{max}() \Rightarrow \text{Output } f(x)
\]

\[
\text{orElse}(0) \Rightarrow \text{Output } g(f(x))
\]

*IntStream is a specialization of Stream for the int primitive.*

See [docs.oracle.com/javase/8/docs/api/java/util/stream/IntStream.html](https://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

\[ \text{IntStream mapToInt(TtoIntFunction<...> mapper)} \]

\[ \text{Input } x \]

\[ \text{Output } f(x) \]

\[ \text{max()} \]

\[ \text{Output } g(f(x)) \]

orElse(0)

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

```
List<Result>          Stream<Result>          IntStream<int>          OptionalInt
```

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

```
resultsList

stream()

max()

orElse(0)
```

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

\[
\text{List} \quad \langle \text{Result}\rangle \\
\text{Stream} \quad \langle \text{Result}\rangle \\
\text{IntStream} \quad \langle \text{int}\rangle \\
\text{OptionalInt}
\]

\[
\text{mapToInt} \left(\text{Result}::\text{getIndex}\right)
\]

\[
\text{resultsList} = [128, 138, 148, 199, 209, 219, 503]
\]

\[
\text{stream()}
\]

\[
\text{mapToInt(\text{Result}::\text{getIndex})}
\]

\[
\text{max()}
\]

\[
\text{orElse(0)}
\]

`mapToInt()` transforms the type of elements it processes.
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

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

Note "fluent" programming style with cascading method calls.

See [en.wikipedia.org/wiki/Fluent_interface](en.wikipedia.org/wiki/Fluent_interface)
End of Java Streams: Intermediate Operations (Part 1)