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

<table>
<thead>
<tr>
<th>Input String to Search</th>
<th>Search Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let's start at the very beginning.</td>
<td>&quot;do&quot;, &quot;re&quot;, &quot;mi&quot;, &quot;fa&quot;, &quot;so&quot;, &quot;la&quot;, &quot;ti&quot;, &quot;do&quot;</td>
</tr>
</tbody>
</table>

We continue to showcase the SimpleSearchStream program

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://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

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

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

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.

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

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

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

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

*ACROSS THE BOARD*

These caveats apply to all “run-to-completion” intermediate operations!
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.

```
map(this::searchForWord)
```

Search Words

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

Overview of the map() Intermediate Operation
Overview of the map() Intermediate Operation

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

List
<String>

Stream
<String>

Stream
<SearchResults>

map(this::searchForWord)

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

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

```java
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](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.

```
IntStream mapToInt(ToIntFunction<...> mapper)
```

- `Input x`
- `Output f(x)`
- `max()`
- `Output g(f(x))`
- `orElse(0)`

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

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

\[\text{Input } x\]

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

\[\text{max()}\]

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

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

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

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

```
IntStream mapToInt(ToIntFunction<…> mapper)
```

```
Input x
```

```
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

List `<Result>`

Stream `<Result>`

IntStream

OptionalInt

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

```
resultsList

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

stream()

mapToInt(Result::getIndex)

max()

orElse(0)
```
Overview of the `mapToInt()` Intermediate Operation

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

```
List<Result>
```

```
Stream<Result>
```

```
IntStream
```

```
OptionalInt
```

```
mapToInt(Result::getIdx)
```

```
resultsList
```

```
max()
```

```
orElse(0)
```

`mapToInt()` transforms the type of elements it processes into primitive ints
Example of applying `mapToInt()` & a mapper function in the SimpleSearchStream program

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

Overview of the `mapToInt()` Intermediate Operation

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

Note “fluent” programming style with cascading method calls.
End of Java Streams
Intermediate Operations
map() & mapToInt()