Understand Java Streams Intermediate Operations filter() & flatMap()
Learning Objectives in this Part of the Lesson

• Understand the structure & functionality of stream aggregate operations
• Intermediate operations
  • map() & mapToInt()
  • filter() & flatMap()

These are both stateless, run-to-completion operations
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

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

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.

\[ \text{Input } x \]

\[ \text{Stream } \text{map}(\text{Function}<\ldots> \text{ mapper}) \]

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

\[ \text{Stream } \text{filter}(\text{Predicate}<\ldots> \text{ pred}) \]

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

\[ R \text{ collect}(\text{Collector}<\ldots> \text{ collector}) \]

The # of output stream elements may be less than the # of input stream elements.
Overview of the filter() Intermediate Operation

- Example of applying filter() & a predicate in the SimpleSearchStream program

```
Stream<SearchResults> = Stream<Map<String, List<String>>>.
map(this::searchForWord)
filter(not(SearchResults::isEmpty))
```

List<String>

Stream<String>

Stream<SearchResults>

Stream<SearchResults>

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

Filter out empty SearchResults.
Overview of the `filter()` Intermediate Operation

- Example of applying `filter()` & a predicate in the SimpleSearchStream program

```
Stream<SearchResults> stream()
    map(this::searchForWord)
    filter(not(SearchResults::isEmpty))
```

*filter() can’t change the type or value of elements it processes*
Overview of the filter() Intermediate Operation

- Example of applying filter() & a predicate in the SimpleSearchStream program

```java
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](en.wikipedia.org/wiki/Fluent_interface)
Overview of the flatMap() Intermediate Operation
Overview of the flatMap() Intermediate Operation

- Returns a stream that replaces each element of this stream w/contents of a mapped stream produced by applying the provided mapping function to each element

This definition sounds like map() at first glance, but there are important differences!

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#flatMap
Overview of the flatMap() Intermediate Operation

- Returns a stream that replaces each element of this stream with contents of a mapped stream produced by applying the provided mapping function to each element.

Stream.of(l1, l2, l3, …, ln)

flatMap(List::stream)

forEach(System.out::println)

The # of output stream elements may differ from the # of input stream elements
Overview of the flatMap() Intermediate Operation

- Returns a stream that replaces each element of this stream w/contents of a mapped stream produced by applying the provided mapping function to each element

```
Stream.of(l1, l2, l3, ..., ln)
.flatMap(List::stream)
.forEach(System.out::println)
```

"Flatten" an array of lists of strings into a stream of strings
Overview of the `flatMap()` Intermediate Operation

- Returns a stream that replaces each element of this stream w/contents of a mapped stream produced by applying the provided mapping function to each element

```
Stream.of(l1, l2, l3, ..., ln)
.flatMap(List::stream)
.forEach(System.out::println)
```

Output `f(x)`

Output `g(f(x))`

Output `...(g(f(x)))`

forEach(System.out::println)

`flatMap()` may transform the type of elements it processes
Overview of the flatMap() Intermediate Operation

- Returns a stream that replaces each element of this stream with contents of a mapped stream produced by applying the provided mapping function to each element.

List<String> l1 = ...;
List<String> l2 = ...;
List<String> l3 = ...;
...
List<String> ln = ...;

Stream
  .of(l1, l2, l3, ..., ln)
  .flatMap(List::stream)
  .forEach(System.out::println);

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12
End of Understand Java Streams
Intermediate Operations
filter() & flatMap()