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

```
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#filter](https://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

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

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

\[
\text{Stream}\lt\text{SearchResults}\gt
\quad \rightarrow \quad \text{map}(\text{this}::\text{searchForWord})
\quad \rightarrow \quad \text{filter}(\text{not}(\text{SearchResults}::\text{isEmpty}))
\quad \rightarrow \quad \text{Stream}\lt\text{SearchResults}\gt
\]

List \lt\text{String}\gt

Search Words

"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

```java
Stream<SearchResults> stream()
  .filter(not(SearchResults::isEmpty))
  .map(this::searchForWord)
  .forEach(System.out::println)
```

**List**
```
(List<String>)
```

**Stream**
```
(Stream<String>)
```

**Stream**
```
(Stream<SearchResults>)
```

**Stream**
```
(Stream<SearchResults>)
```

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

- 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](http://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 with 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!

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

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

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 with the 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)

```
…
```

**flatMap(List::stream)**

**Output**: g(f(x))

```
…
```

**forEach(System.out::println)**

**Output**: …(g(f(x)))

“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 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)
```

Output f(x)

```
Stream<List<String>>
array<List<String>>
…
Stream<String>
```

Output g(f(x))

```
forEach(System.out::println)
```

Output …(g(f(x)))

flatMap() may transform the type of elements it processes
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);

End of Java Streams
Intermediate Operations
filter() & flatMap()