Java Streams: Terminal Operations

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
  • Terminal operations
Learning Objectives in this Part of the Lesson

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

These operations also apply to both sequential & parallel streams
Learning Objectives in this Part of the Lesson

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

We continue to showcase the SimpleSearchStream program

Input x

\[ \map{\text{this::searchForWord}} \]

Output \( f(x) \)

\[ \text{filter(not(SearchResults::isEmpty))} \]

Output \( g(f(x)) \)

\[ \text{collect(toList())} \]
Overview of Terminal Operations
Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result

```java
Stream
  .of("horatio",
      "laertes",
      "Hamlet",
      ...
  )
  .filter(s -> toLowerCase(s.charAt(0)) == 'h')
  .map(this::capitalize)
  .sorted()
  .forEach(System.out::println);
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12)
Overview of Common Stream Terminal Operations

• Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  • No value at all
  • e.g., forEach() & forEachOrdered()

*forEach() only has side-effects!*
Every stream finishes with a terminal operation that yields a non-stream result, e.g.

- No value at all
  - e.g., forEach() & forEachOrdered()

```java
Stream.of("horatio",
    "laertes",
    "Hamlet",
    ...
).filter(s -> toLowerCase(s.charAt(0)) == 'h')
  .map(this::capitalize)
  .sorted()
  .forEach(System.out::println);
```

Print each character in Hamlet that starts with ‘H’ or ‘h’ in consistently capitalized & sorted order.
Overview of Common Stream Terminal Operations

- Every stream finishes with a terminal operation that yields a non-stream result, e.g.
  - No value at all
  - The result of a reduction operation
    - e.g., collect() & reduce()

See docs.oracle.com/javase/tutorial/collectionsstreams/reduction.html
Every stream finishes with a terminal operation that yields a non-stream result, e.g.
- No value at all
- The result of a reduction operation
  - e.g., collect() & reduce()

*collect() & reduce() terminal operations work seamlessly with parallel streams.*

See [docs.oracle.com/javase/tutorial/collections_streams/parallelism.html](http://docs.oracle.com/javase/tutorial/collections_streams/parallelism.html)
Overview of the collect() Terminal Operation

- A terminal operation also triggers all the intermediate operation processing
Overview of the collect() Terminal Operation
Overview of the collect() Terminal Operation

This terminal operation uses a collector to perform a reduction on the elements of its input stream & returns the results of the reduction.

Input x

Stream map(Function<…> mapper)

Output f(x)

Stream filter(Predicate<…> pred)

Output g(f(x))

R collect(Collector<…> collector)

Output h(g(f(x)))

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

- Overview of the collect() terminal operation

```
List <String>

Stream <String>

Stream <SearchResults>

Stream <SearchResults>

List <SearchResults>
```

**Search Words**

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

- `stream()`
- `map(this::searchForWord)`
- `filter(not(SearchResults::isEmpty))`
- `collect(toList())`

Triggers intermediate operation processing.
The list returned from collect() presents search results in “encounter order”

Create a list of non-empty SearchResults.

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

Overview of the collect() Terminal Operation

- Overview of the collect() terminal operation

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

stream()
map(this::searchForWord)
filter(not(SearchResults::isEmpty))
collect(toList())
End of Java Streams: Terminal Operations