Understand 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 terminal operations
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of stream 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 terminal operations

Let's start at the very beginning.

We continue to showcase the SimpleSearchStream program

<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>

Input x

map(this::searchForWord)

Output f(x)

filter(not(SearchResults::isEmpty))

Output g(f(x))

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.

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

**Diagram**
- **Input x**
- **Intermediate operation (behavior f)**
- **Output f(x)**
- **Intermediate operation (behavior g)**
- **Output g(f(x))**
- **Terminal operation (behavior h)**
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()`

```java
forEach() & forEachOrdered() only have 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/collections/streams/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 ("lazy") intermediate operation processing.
End of Understand Java Streams Terminal Operations