Understanding Java Streams Short-Circuit Aggregate 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

• Understand the Java stream “short-circuit” aggregate operations

\[
\text{Input } x \quad \xrightarrow{\text{takeWhile (predicate)}} \quad \text{Output } f(x) \quad \xrightarrow{\text{limit (maxSize)}} \quad \text{Output } g(f(x)) \quad \xrightarrow{\text{findAny()}} \quad \text{Output}
\]
Java Streams Short-Circuit Operations
Java Streams Short-Circuit Operations

• An aggregate operation *may* process all elements in a stream
Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
- `map()` processes all of the elements in its input stream

```
Input x
Stream map(Function<...> mapper)
Output f(x)
Stream takeWhile(Predicate<...> p)
Output g(f(x))
Stream limit(long maxSize)
Output g(f(x))
Optional findFirst()
```
Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
  - map() processes all of the elements in its input stream
  - Unless a behavior throws an exception.

[Diagram]

Input x

Stream `map(Function<...> mapper)`

Output `f(x)`

Stream `takeWhile(Predicate<...> p)`

Output `g(f(x))`

Stream `limit(long maxSize)`

Output `g(f(x))`

Optional `findFirst()`

See [vanilla-java.github.io/2016/06/21/Reviewing-Exception-Handling.html](vanilla-java.github.io/2016/06/21/Reviewing-Exception-Handling.html)
Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
  - `map()` processes all of the elements in its input stream
  - “Short-circuit” operations halt further processing after reaching their condition

Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
  - `map()` processes all of the elements in its input stream
- “Short-circuit” operations halt further processing after reaching their condition
  - `takeWhile()`
    - A short-circuit intermediate operation that returns a stream consisting of a subset of elements taken from this stream that match the given predicate

See [docs.oracle.com/javase/9/docs/api/java/util/stream/Stream.html#takeWhile](docs.oracle.com/javase/9/docs/api/java/util/stream/Stream.html#takeWhile)
Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
  - `map()` processes all of the elements in its input stream
  - “Short-circuit” operations halt further processing after reaching their condition
    - `takeWhile()`
    - `limit()`
- A short-circuit intermediate operation that causes a stream to operate on a reduced size

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#limit](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#limit)
Java Streams Short-Circuit Operations

- An aggregate operation *may* process all elements in a stream, e.g.
  - `map()` processes all of the elements in its input stream
  - “Short-circuit” operations halt further processing after reaching their condition
    - `takeWhile()`
    - `limit()`
    - `findFirst()`, `findAny()`, `anyMatch()`, `allMatch()`, & `noneMatch()`
  - Short-circuit terminal operations can finish before traversing all elements in the underlying stream

See [dzone.com/articles/collectors-part-1-%E2%80%93-reductions](dzone.com/articles/collectors-part-1-%E2%80%93-reductions)
End of Understanding Java Streams Short-Circuit Aggregate Operations