Overview of 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

Intermediate operation (Behavior f)

Output f(x)

Intermediate operation (Behavior g)

Output g(f(x))

Terminal operation (Behavior h)
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of stream terminal operations

We continue to showcase the “Hamlet” program

```
Array of names
of("horatio", "laertes", "Hamlet", …)

Stream of names
filter(s->toLowerCase(s.charAt(0))…)

Stream of names starting with ‘h’
map(this::capitalize)

Stream of capitalized names
sorted()

Stream of sorted names
forEach()
```
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);
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12)
Every stream finishes with a terminal operation that yields a non-stream result, e.g.

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

`forEach()` & `forEachOrdered()` only have side-effects!
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
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
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
- An Optional or boolean value
  - e.g., `findAny()`, `findFirst()`, `noneMatch()`, etc.

These terminal operations are "short-circuiting"

```java
List<String> countries = Arrays.asList("france", "india", "china", "usa");
print(countries.stream()
  .filter(country -> country.contains("i"))
  .findFirst().get());
print(countries.stream()
  .filter(country -> country.contains("i"))
  .findAny().get());
print(countries.stream()
  .noneMatch(country -> country.contains("z")));```

See [dzone.com/articles/collectors-part-1-%E2%80%93-reductions](http://dzone.com/articles/collectors-part-1-%E2%80%93-reductions)
Overview of the collect() Terminal Operation

- A terminal operation also triggers all the ("lazy") intermediate operation processing
End of Overview of Java Streams Terminal Operations