Java Streams: the reduce() Terminal Operation

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 common terminal operations, e.g.
  - forEach()
  - collect()
  - reduce()

```java
void runCollectReduce() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap.values().stream().reduce(0L, Long::sum);
}
```

We showcase reduce() using the Hamlet program

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12)
A Stream Terminal Operation That Returns a Primitive
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap
                           .values()
                           .stream()
                           .reduce(0L,
                                    Long::sum);
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce)
The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long> matchingCharactersMap =
        ...
            .collect
                (groupingBy
                    (identity(),
                        TreeMap::new,
                        summingLong
                            (String::length)));
```

Create a map associating the names of Hamlet characters with their name lengths.

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12)
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long>
        matchingCharactersMap = ...

    long sumOfNameLengths =
        matchingCharactersMap
            .values()
            .stream()
            .reduce(0L, 
                    Long::sum);
```

Convert the map’s values list into a stream of long values.
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long>
    matchingCharactersMap = ...

    long sumOfNameLengths =
    matchingCharactersMap
    .values()
    .stream()
    .reduce(0L,
            Long::sum);
```

Sum up the lengths of all character names in Hamlet.
The reduce() terminal operation returns a primitive value. 0 is the “identity,” i.e., the initial value of the reduction & the default result if there are no elements in the stream.
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap.values().stream().reduce(0L, Long::sum);
}
```

This method reference is an “accumulator,” which is a stateless function that combines two values into a single (immutable) “reduced” value.

See [docs.oracle.com/javase/8/docs/api/java/lang/Long.html#sum](docs.oracle.com/javase/8/docs/api/java/lang/Long.html#sum)
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation returns a primitive value

```java
void runCollectReduce1() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap.values().stream()
        .reduce(0L, (x, y) -> x + y);
```

A lambda expression could also be used here.

See [stackoverflow.com/a/24493905](https://stackoverflow.com/a/24493905)
A Stream Terminal Operation That Returns a Primitive

- The three parameter “map/reduce” version of reduce() is used along with parallel streams.

```java
void runCollectMapReduce() {
    List<String> characterList = ...

    long sumOfNameLengths =
        characterList
            .parallelStream()
            .reduce(0L,
                    (sum, s) ->
                        sum + s.length(),
                    Long::sum);
}
```

See [www.youtube.com/watch?v=oWIWEKNM5Aw](https://www.youtube.com/watch?v=oWIWEKNM5Aw)
A Stream Terminal Operation That Returns a Primitive

- The three parameter "map/reduce" version of reduce() is used along with parallel streams

```java
void runCollectMapReduce() {
    List<String> characterList = ...

    long sumOfNameLengths = characterList
        .parallelStream()
        .reduce(0L,
               (sum, s) -> sum + s.length(),
               Long::sum);
```

Convert the list into a parallel stream.
A Stream Terminal Operation That Returns a Primitive

- The three parameter “map/reduce” version of reduce() is used along with parallel streams

```java
void runCollectMapReduce() {
    List<String> characterList = ...;

    long sumOfNameLengths =
        characterList
            .parallelStream()
            .reduce(0L,
                    (sum, s) ->
                        sum + s.length(),
                    Long::sum);
}
```

Perform a reduction on the stream with an initial value of 0.

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce](https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#reduce)
The three parameter “map/reduce” version of reduce() is used along with parallel streams.

```java
void runCollectMapReduce() {
    List<String> characterList = ...

    long sumOfNameLengths =
        characterList
            .parallelStream()
            .reduce(0L,
                    (sum, s) ->
                        sum + s.length(),
                    Long::sum);
```

This lambda expression is an accumulator that performs the “map” operation.
A Stream Terminal Operation That Returns a Primitive

- The three parameter “map/reduce” version of reduce() is used along with parallel streams

```java
void runCollectMapReduce() {
    List<String> characterList = ...

    long sumOfNameLengths =
        characterList
            .parallelStream()
            .reduce(0L,
                    (sum, s) ->
                    sum + s.length(),
                    Long::sum);
}
```

This method reference performs the “reduce” operation.

See [docs.oracle.com/javase/8/docs/api/java/lang/Long.html#sum](docs.oracle.com/javase/8/docs/api/java/lang/Long.html#sum)
A Stream Terminal Operation That Returns a Primitive

- The sum() terminal operation avoids the need to use reduce()

```java
void runCollectReduce2() {
    Map<String, Long>
        matchingCharactersMap =
            ...

    long sumOfNameLengths =
        matchingCharactersMap
            .values()
            .stream()
            .mapToLong(Long::longValue)
            .sum()
}
```

See docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#sum
The sum() terminal operation avoids the need to use reduce()

```java
void runCollectReduce2() {
    Map<String, Long>
    matchingCharactersMap = ...

    long sumOfNameLengths =
        matchingCharactersMap
            .values()
            .stream()
            .mapToLong(Long::longValue)
            .sum();
}
```

Convert the map into a stream of long values.
A Stream Terminal Operation That Returns a Primitive

- The sum() terminal operation avoids the need to use reduce()

```java
void runCollectReduce2() {
    Map<String, Long>
        matchingCharactersMap = ...

        long sumOfNameLengths =
            matchingCharactersMap
                .values()
                .stream()
                .mapToLong(Long::longValue)
                .sum();
}
```

Map the stream of Long objects into a stream of long primitives.

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#mapToLong
A Stream Terminal Operation That Returns a Primitive

- The `sum()` terminal operation avoids the need to use `reduce()`

```java
void runCollectReduce2() {
    Map<String, Long>
        matchingCharactersMap = ...

    long sumOfNameLengths =
        matchingCharactersMap
            .values()
            .stream()
            .mapToLong(Long::longValue)
            .sum()
}
```

*Sum the stream of long primitives into a single result.*

See [docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#sum](https://docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#sum)
A Stream Terminal Operation That Returns a Primitive

• The collect() terminal operation can also be used to return a primitive value

```java
void runCollectReduce3() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap
        .values()
        .stream()
        .collect(
            summingLong
            (Long::longValue));
}
The `collect()` terminal operation can also be used to return a primitive value.

```java
void runCollectReduce3() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths =
        matchingCharactersMap.values().stream()
            .collect(summingLong(Long::longValue));
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect)
A Stream Terminal Operation That Returns a Primitive

- The `collect()` terminal operation can also be used to return a primitive value.

```java
void runCollectReduce3() {
    Map<String, Long> matchingCharactersMap = ...

    long sumOfNameLengths = matchingCharactersMap.values().stream().collect(summingLong(Long::longValue));
}
```

Return a collector that produces the sum of a long-value function applied to input elements.

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#summingLong](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#summingLong)
End of Java Streams: the reduce() Terminal Operation