Understand the Java Streams
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()

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/douglascraigschmidt/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 typically 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
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation typically 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
A Stream Terminal Operation That Returns a Primitive Value

- The reduce() terminal operation typically 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 typically 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 typically 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.

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

    long sumOfNameLengths =
        matchingCharactersMap
            .values()
            .stream()
            .reduce(0L,
            Long::sum);
```
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation typically 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
A Stream Terminal Operation That Returns a Primitive

- The reduce() terminal operation typically 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](http://www.youtube.com/watch?v=oWIWEKNM5Aw)
The three parameter “map/reduce” version of reduce() is used along with parallel streams.

Generate a consistently capitalized & sorted list of names of Hamlet characters starting with the letter 'h'.

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

    long sumOfNameLengths = characterList
        .parallelStream()
        .reduce(0L,
               (sum, s) ->
                   sum + s.length(),
               Long::sum);
}
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
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.*
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](http://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](http://docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.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();
```

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](http://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

- `collect()` 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 earlier lesson on “Java Streams: the `collect()` Terminal Operation”
A Stream Terminal Operation That Returns a Primitive

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

- collect() 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](docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#summingLong)
A Stream Terminal Operation That Returns a Primitive

- `reduce()` can also be used to return a non-primitive value

```java
void streamReduceConcat
   (boolean parallel) {
   ...
   Stream<String> wordStream = allWords.stream();
   ...
   String words = wordStream
      .reduce(new String(),
             (x, y) -> x + y);

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex17
reduce() can also be used to return a non-primitive value.

A Stream Terminal Operation That Returns a Primitive

```java
void streamReduceConcat(boolean parallel) {
  Stream<String> wordStream = allWords.stream();
  String words = wordStream
                  .reduce(wordStream)
                  .reduce(new String(), ((x, y) -> x + y));
}
```

reduce() creates a string containing all the concatenated words in a stream.

See javarevisited.blogspot.com/2015/01/3-examples-to-concatenate-string-in-java.html
A Stream Terminal Operation That Returns a Primitive

• reduce() can also be used to return a non-primitive value

```java
void streamReduceConcat (boolean parallel) {
  ...
  Stream<String> wordStream = allWords.stream();
  ...
  String words = wordStream .reduce(new String(),
                                         (x, y) -> x + y);
}
```

This simple fix is inefficient due to string concatenation overhead

See upcoming lesson on “Java Parallel Stream Internals: Combining Results (Part 2)”
End of Understand the Java Streams reduce() Terminal Operation