Contrasting the Java Streams reduce() & collect() Terminal Operations

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

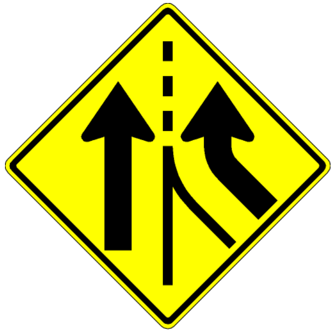
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

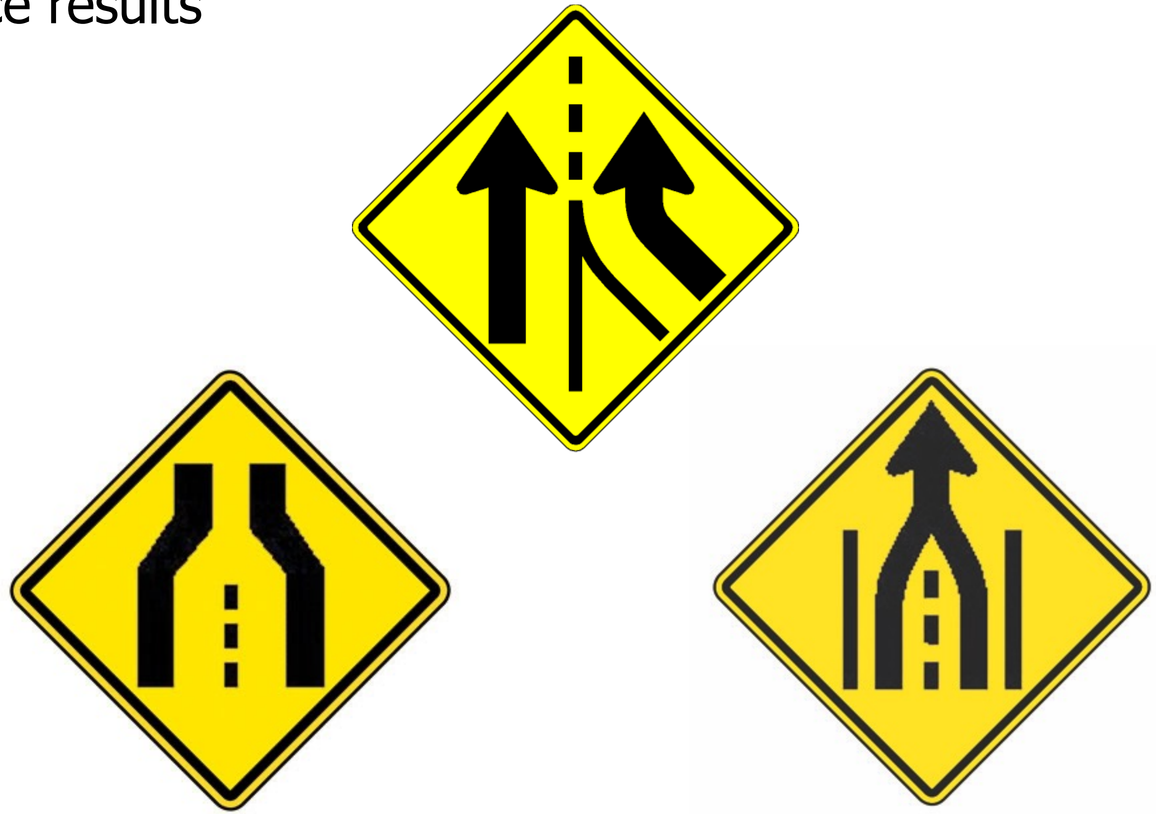
- Understand common terminal operations, e.g.
  - forEach()
  - collect()
  - reduce()
  - Contrasting reduce() & collect()



Contrasting the reduce() & collect() Terminal Operations

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways



These differences are important for parallel streams (covered later)

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value

An immutable value cannot be modified once it's created

See docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value

```
long factorial(long n) {
  return LongStream
    .rangeClosed(1, n)
    .reduce(1, (a, b) -> a * b);
}
```

Compute the product of all positive integers <= to n (denoted as n!)

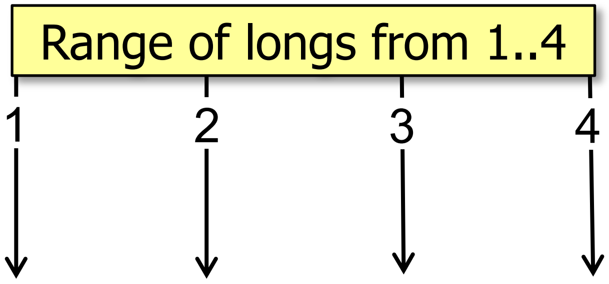
See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex16

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value

```
long factorial(long n) {
  return LongStream
    .rangeClosed(1, n)
    .reduce(1, (a, b) -> a * b);
}
```

Generate a range of primitive long values from 1 to n (inclusive)

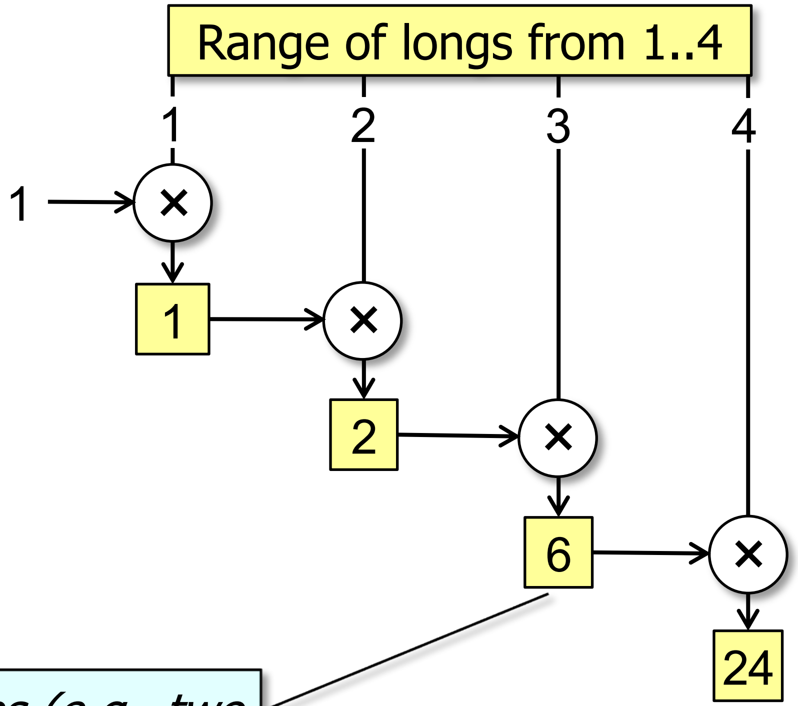


See docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#rangeClosed

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value

```
long factorial(long n) {
  return LongStream
    .rangeClosed(1, n)
    .reduce(1, (a, b) -> a * b);
}
```

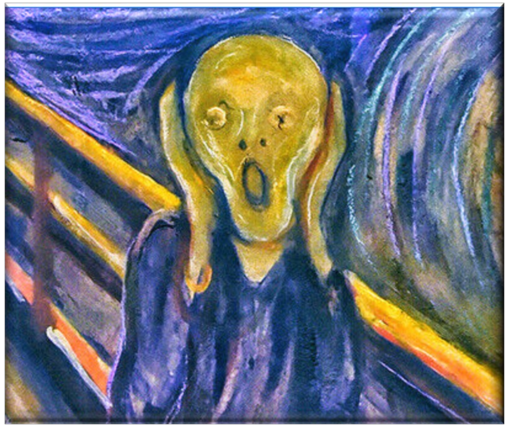


reduce() combines two immutable values (e.g., two long, int, etc.) & produces a new immutable value

See docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#reduce

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
    - Chaos & insanity will result if reduce() is used on mutable objects..



```
void buggyStreamReduce3a
                (boolean parallel) {
  ...
  Stream<String> wordStream =
    allStrings.stream();

  if (parallel)
    wordStream.parallel();

  String words = wordStream
    .reduce(new StringBuilder(),
            StringBuilder::append,
            StringBuilder::append)
    .toString();
```

See upcoming lesson on "*Java Parallel Streams Internals: Combining Results (Part 2)*"

Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

It gathers elements from the stream into a specified mutable results container

See greenteapress.com/thinkapjava/html/thinkjava011.html

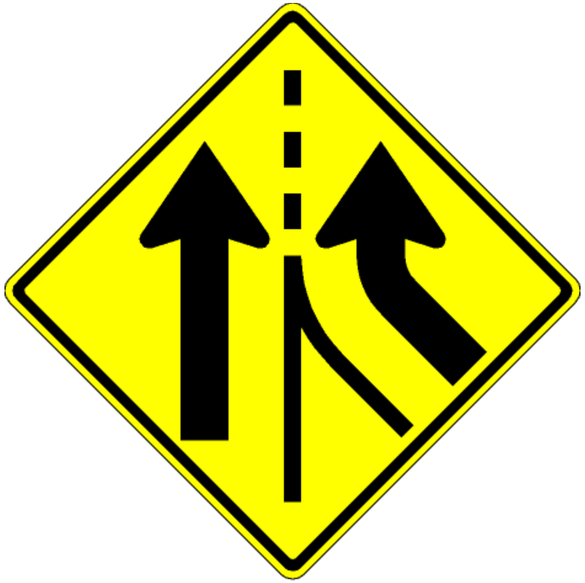
Contrasting the reduce() & collect() Terminal Operations

- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
  getInput(sSHAKESPEARE,sSPLIT_WORDS)
  .stream()

  .map(string ->
       string.toString()
             .toLowerCase())

  .collect(toCollection(TreeSet::new));
```



See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex36

Contrasting the reduce() & collect() Terminal Operations

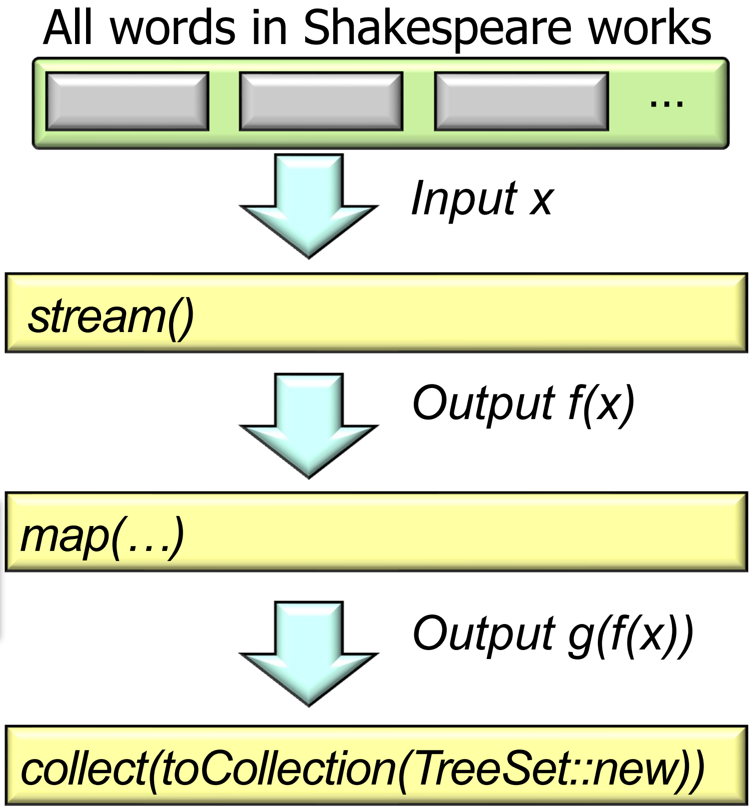
- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
  getInput(sSHAKESPEARE, sSPLIT_WORDS)
  .stream()

  .map(string ->
       string.toString()
             .toLowerCase())

  .collect(toCollection(TreeSet::new));
```

Create a set of all unique words in Shakespeare



All words in Shakespeare works
...
Input x
stream()
Output f(x)
map(...)
Output g(f(x))
collect(toCollection(TreeSet::new))

Contrasting the reduce() & collect() Terminal Operations

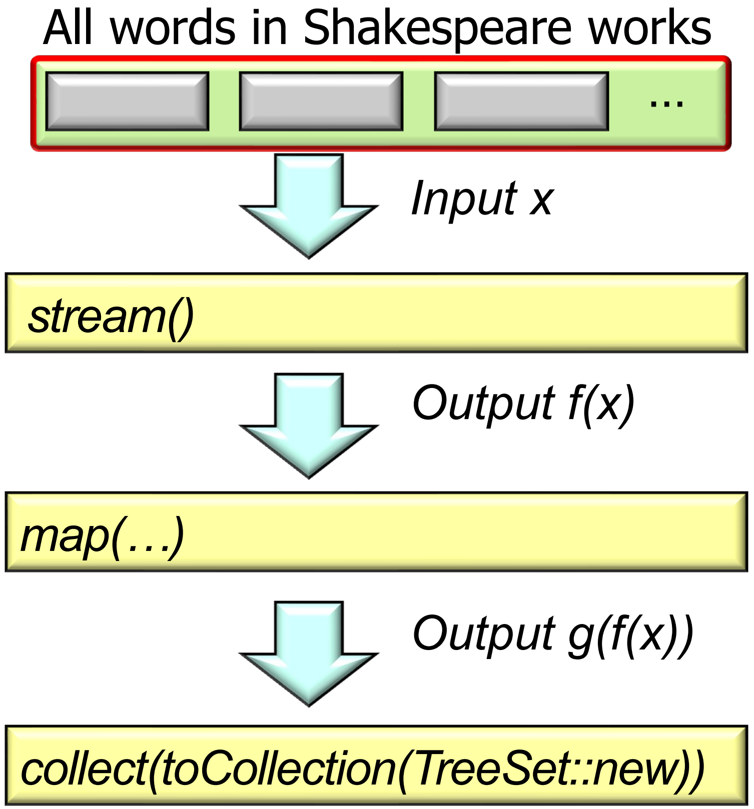
- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
  getInput(sSHAKESPEARE,sSPLIT_WORDS)
  .stream()

  .map(string ->
       string.toString()
             .toLowerCase())

  .collect(toCollection(TreeSet::new));
```

Get list of all words in Shakespeare


All words in Shakespeare works
…
Input x
stream()
Output f(x)
map(…)
Output g(f(x))
collect(toCollection(TreeSet::new))

Contrasting the reduce() & collect() Terminal Operations

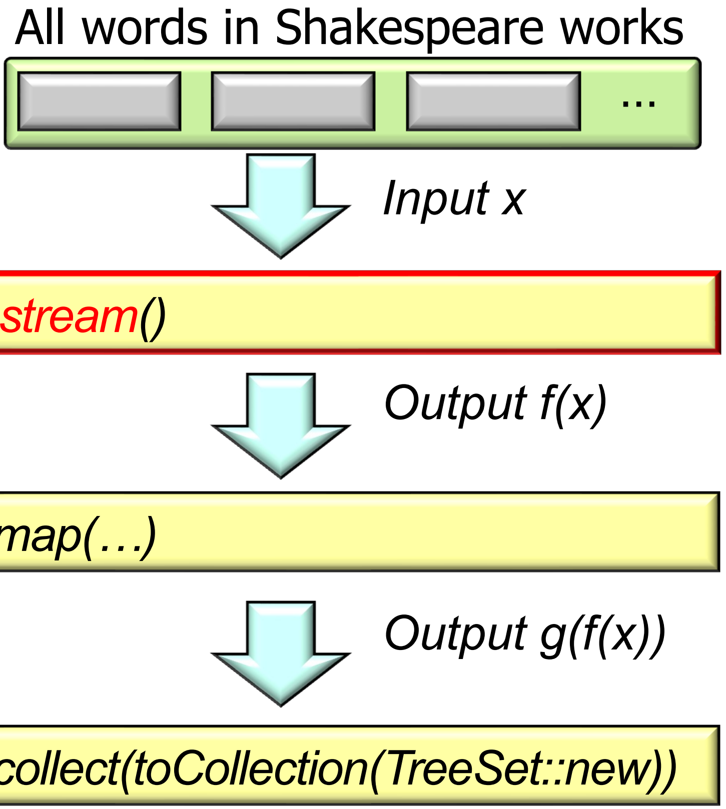
- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
  getInput(sSHAKESPEARE,sSPLIT_WORDS)
  .stream()

  .map(string ->
       string.toString()
             .toLowerCase())

  .collect(toCollection(TreeSet::new));
```

Convert list into stream


All words in Shakespeare works
...
Input x
stream()
Output f(x)
map(...)
Output g(f(x))
collect(toCollection(TreeSet::new))

Contrasting the reduce() & collect() Terminal Operations

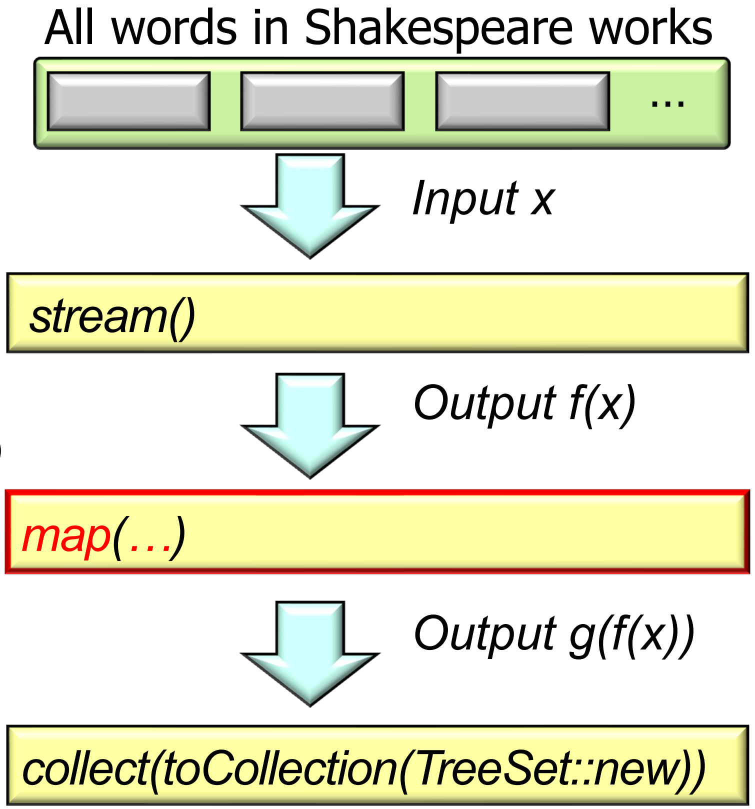
- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
   getInput(sSHAKESPEARE,sSPLIT_WORDS)
   .stream()

   .map(string ->
        string.toString()
              .toLowerCase())

   .collect(toCollection(TreeSet::new));
```

Lower case all words


All words in Shakespeare works
…
Input x
stream()
Output f(x)
map(…)
Output g(f(x))
collect(toCollection(TreeSet::new))

Contrasting the reduce() & collect() Terminal Operations

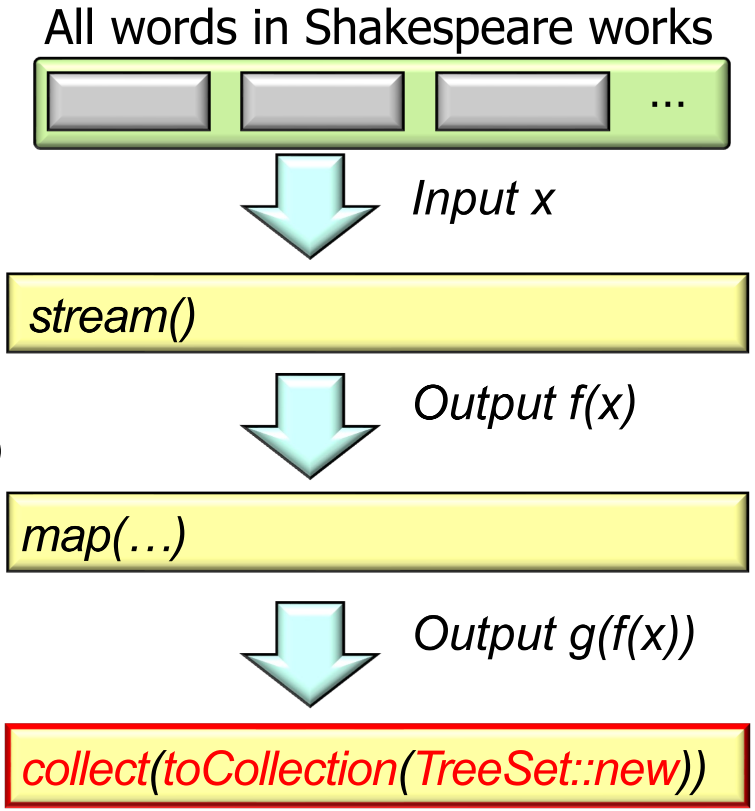
- Terminal operations produce results in different ways, e.g.
  - reduce() creates an immutable value
  - collect() mutates an existing object

```
Set<String> uniqueWords =
  getInput(sSHAKESPEARE,sSPLIT_WORDS)
  .stream()

  .map(string ->
       string.toString()
             .toLowerCase())

  .collect(toCollection(TreeSet::new));
```

Collect into a TreeSet


All words in Shakespeare works
…
Input x
stream()
Output f(x)
map(…)
Output g(f(x))
collect(toCollection(TreeSet::new))

toCollection() creates a TreeSet container & accumulates stream elements into it

End of Contrasting the Java Streams reduce() & collect() Terminal Operations