Java Streams: Contrasting the 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 `reduce()` & `collect()`
Contrasting reduce() & collect()

- Terminal operations produce results in different ways

These differences are important for parallel streams (covered later)
Contrasting `reduce()` & `collect()`

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

See [docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html](docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html)
Contrasting `reduce()` & `collect()`

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

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

Contrasting `reduce()` & `collect()`

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

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

See [docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#rangeClosed](http://docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#rangeClosed)
Terminal operations produce results in different ways, e.g.

- `reduce()` creates an immutable value

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

Contrasting `reduce()` & `collect()`: The range of longs from 1..4:

1 2 3 4

1 × 2

2 × 3

6 × 4

24

reduce() combines two immutable values (e.g., long or Long) & produces a new one

See docs.oracle.com/javase/8/docs/api/java/util/stream/LongStream.html#reduce
Contrasting reduce() & collect()

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

See greenteapress.com/thinkapjava/html/thinkjava011.html
Contrasting `reduce()` & `collect()`

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

```java
Set<CharSequence> uniqueWords =
    getInput(sSHAKESPEARE, "\s+")
    .stream()
    .map(charSeq ->
        charSeq.toString()
        .toLowerCase()
    )
    .collect(toCollection(HashSet::new));
```
Contrasting reduce() & collect()

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

```java
Set<CharSequence> uniqueWords =
    getInput(sSHAKESPEARE, "\s+")
    .stream()
    .map(charSeq -> charSeq.toString()
          .toLowerCase())
    .collect(toCollection(HashSet::new));
```

Output $f(x)$

Input $x$

Output $g(f(x))$

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex14](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex14)
Contrasting `reduce()` & `collect()`

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

```java
Set<CharSequence> uniqueWords =
    getInput(sSHAKESPEARE, "\s+")
    .stream()
    .map(charSeq ->
        charSeq.toString()
            .toLowerCase())
    .collect(toCollection(HashSet::new));
```

- All words in Shakespeare works
- `stream()`
- `map(...)`
- `collect(toCollection(HashSet::new))`
Contrasting reduce() & collect()

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

Set<CharSequence> uniqueWords =
getInput(sSHAKESPEARE),
"\\s+")
.stream()
.map(charSeq ->
  charSeq.toString()
    .toLowerCase())
.collect(toCollection(HashSet::new));
Contrasting reduce() & collect()

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

Set<CharSequence> uniqueWords =
getInput(sSHAKESPEARE, "\s+")
.stream()
  .map(charSeq -> charSeq.toString().toLowerCase())
  .collect(toCollection(HashSet::new));
Contrasting `reduce()` & `collect()`

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

```java
Set<CharSequence> uniqueWords =
    getInput(sSHAKESPEARE, "\\s+")
    .stream()
    .map(charSeq ->
        charSeq.toString()
        .toLowerCase())
    .collect(toCollection(HashSet::new));
```

- All words in Shakespeare works
- `stream()`
- `map(…)`
- `collect(toCollection(HashSet::new))`

- `reduce()` creates an immutable value
- `collect()` mutates an existing value

---

**Collect into a HashSet**

- `toCollection()` creates a HashSet container & accumulates stream elements into it
End of Java Streams: Contrasting reduce() & collect()