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

See [docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html](https://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

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

Contrasting the reduce() & collect() Terminal Operations

- 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

Generate a range of primitive long values from 1 to n (inclusive)
Contrasting the reduce() & collect() Terminal Operations

- 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#reduce
Contrasting the reduce() & collect() Terminal Operations

- 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 the reduce() & collect() Terminal Operations

- 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(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 value

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

Input x

Output f(x)

Output g(f(x))

Create a set of all the unique words in Shakespeare’s works

See [github.com/douglas craigschmidt/LiveLessons/tree/master/Java8/ex14](https://github.com/douglas craigschmidt/LiveLessons/tree/master/Java8/ex14)
Contrasting the reduce() & collect() Terminal Operations

- 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(TreeSet::new));
```

Input $x$

Output $f(x)$

Output $g(f(x))$

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

Get list of all words in Shakespeare

All words in Shakespeare works
Set<CharSequence> uniqueWords =
    getInput(sSHAKESPEARE),
    "\s+"
    .stream()
    .map(charSeq ->
        charSeq.toString()
        .toLowerCase())
    .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 value

All words in Shakespeare works

Input x

Output f(x)

Output g(f(x))

\textit{Convert list into stream}
Contrasting the reduce() & collect() Terminal Operations

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

\[
\text{Set<CharSequence> uniqueWords =}
\begin{align*}
& \text{getInput(sSHAKESPEARE),} \\
& \quad "\\s+"
\end{align*}
\]
\[
\begin{align*}
& .\text{stream()} \\
& .\text{map(charSeq -> charSeq.toString())} \\
& \quad .\text{toLowerCase()} \\
& .\text{collect(toCollection(TreeSet::new))}
\end{align*}
\]

• Terminal operations produce results in different ways, e.g.
  • reduce() creates an immutable value
  • collect() mutates an existing value
Contrasting the `reduce()` & `collect()` Terminal Operations

• 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(TreeSet::new));
```

Terminal operations produce results in different ways, e.g.

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

Contrasting the `reduce()` & `collect()` Terminal Operations

- `input x`
- `Output f(x)`
- `Output g(f(x))`
- `Collect into a TreeSet`

`toCollection()` creates a TreeSet container & accumulates stream elements into it
End of Contrasting the Java Streams reduce() & collect() Terminal Operations