Contrast 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
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);
}
```

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](https://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

```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
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(HashSet::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(HashSet::new));
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex14](https://github.com/douglascraigschmidt/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

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

All words in Shakespeare works

```
Input x
```

```
stream()
```

```
Output f(x)
```

```
map(...)
```

```
Output g(f(x))
```

```
collect(toCollection(HashSet::new))
```

Get list of all words in Shakespeare
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(HashSet::new));

Output f(x)

Input x

Output g(f(x))

Convert list into stream

All words in Shakespeare works
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())
    .map(charSeq -> charSeq.toLowerCase())
    .collect(toCollection(HashSet::new));
```

• 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))`
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(HashSet::new));
```

- ToCollection() creates a HashSet container & accumulates stream elements into it
End of Contrast the Java Streams reduce() & collect() Terminal Operations