Java Streams: the collect() 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
We showcase collect() using the Hamlet program

```java
void runCollectTo*() {
    List<String> characters = Arrays.asList("horatio", "laertes", "Hamlet", ...);

    ...<String> results = characters
        .stream()
        .filter(s -> toLowerCase(...) == 'h')
        .map(this::capitalize)
        .sorted()
        .collect(...); ...
}
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12
A Stream Terminal Operation That Returns Collections
A Stream Terminal Operation That Returns Collections

- The collect() terminal operation typically returns a collection.

```java
typical returns a collection
void runCollectTo*() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);

    ...<String> results =
        characters
            .stream()
            .filter(s ->
                toLowerCase(...) =='h')
            .map(this::capitalize)
            .sorted()
            .collect(...); ... 
```

See [www.concretepage.com/java/jdk-8/java-8-stream-collect-example](http://www.concretepage.com/java/jdk-8/java-8-stream-collect-example)
A Stream Terminal Operation That Returns Collections

The collect() terminal operation typically returns a collection.

Many variants of collect() are showcased in this example.

```java
void runCollectTo*() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);

    ...<String> results =
        characters
            .stream()
            .filter(s ->
                toLowerCase(...) == 'h')
            .map(this::capitalize)
            .sorted()
            .collect(...); ...
```

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12)
The `collect()` terminal operation typically returns a collection:

```java
void runCollectTo*() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);

    ...<String> results =
        characters
            .stream()
            .filter(s ->
                toLowerCase(...) == 'h')
            .map(this::capitalize)
            .sorted()
            .collect(...); ...
```

Create & process a stream consisting of characters from the play "Hamlet"
The collect() terminal operation typically returns a collection.

```java
void runCollectTo*() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);

    // Performs a mutable reduction on all elements of this stream
    // using some type of collector

    ...<String> results =
        characters.stream()
            .filter(s -> toLowerCase(...) == 'h')
            .map(this::capitalize)
            .sorted()
            .collect(...); ...
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect)
The collect() terminal operation typically returns a collection

```java
void runCollectTo*() {
    List<String> characters = Arrays.asList("horatio",
                                           "laertes",
                                           "Hamlet", ...);

    ...<String> results =
    characters
        .stream()
        .filter(s ->
            toLowerCase(...) == 'h')
        .map(this::capitalize)
        .sorted()
        .collect(...); ...
```

A collector performs reduction operations, e.g., summarizing elements according to various criteria, accumulating elements into various types of collections, etc.

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html](docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html)
A Stream Terminal Operation That Returns Collections

- The collect() terminal operation typically returns a collection

```java
void runCollectToList() {
    List<String> characters =
        Arrays.asList("horatio", "laertes", "Hamlet, ...");

    List<String> results =
        characters
            .stream()
            .filter(s ->
                toLowerCase(...) == 'h')
            .map(this::capitalize)
            .sorted()
            .collect(toList()); ...
}
```

Collect results into an ArrayList, which can contain duplicates.

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#toList
A Stream Terminal Operation That Returns Collections

- The `collect()` terminal operation typically returns a collection

```java
void runCollectToList() {
    List<String> characters =
        Arrays.asList("horatio", "laertes", "Hamlet, ...); 
    List<String> results =
        characters.stream()
            .filter(s ->
                s.toLowerCase() == 'h')
            .map(this::capitalize)
            .sorted()
            .collect(toList()); ...
}
```

*Using `collect()` is less error-prone than `forEach()` since initialization is implicit & it’s inherently thread-safe*

See earlier lessons on “Java Streams: the `forEach()` Terminal Operation
A Stream Terminal Operation That Returns Collections

- The `collect()` terminal operation typically returns a collection

```java
void runCollectToSet() {
    List<String> characters = Arrays.asList("horatio", "laertes", "Hamlet", ...);
    Set<String> results = characters
        .stream()
        .filter(s -> toLowerCase(...) == 'h')
        .map(this::capitalize)
        .collect(toSet());
}
```

Collect the results into a HashSet, which can contain no duplicates.

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#toSet
The collect() terminal operation typically returns a collection

```java
void runCollectToMap() {
    List<String> characters = Arrays.asList("horatio", "laertes", "Hamlet", ...);
    Map<String, Integer> results = characters.stream()
        .filter(s -> s.toLowerCase() == 'h')
        .map(this::capitalize)
        .collect(toMap(identity(), String::length, Integer::sum));
    ...
}
```

Collect results into a HashMap, along with the length of (merged duplicate) entries.

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#toMap
A Stream Terminal Operation That Returns Collections

- The collect() terminal operation typically returns a collection

```java
void runCollectGroupingBy() {
    List<String> characters = Arrays.asList("horatio", "laertes", "Hamlet", ...);
    Map<String, Long> results = ...
    .collect(groupingBy
        (identity(),
            TreeMap::new,
            summingLong
                (String::length)));
    ...}
```

Collect the results into a TreeMap by grouping elements according to name (key) & name length (value).

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#groupingBy
The `collect()` terminal operation typically returns a collection.

```java
doRunCollectGroupingBy() {  
    List<String> characters =  
        Arrays.asList("horatio",  
                      "laertes",  
                      "Hamlet", ...);  
    Map<String, Long> results =  
        ...  
        .collect  
        (groupingBy  
         (identity(),  
          TreeMap::new,  
          summingLong  
          (String::length)));  
    ...  
}
```

groupingBy() partitions a stream via a "classifier" function (identity() always returns its input argument).

See [docs.oracle.com/javase/8/docs/api/java/util/function/Function.html#identity](docs.oracle.com/javase/8/docs/api/java/util/function/Function.html#identity)
A Stream Terminal Operation That Returns Collections

- The `collect()` terminal operation typically returns a collection

```
void runCollectGroupingBy() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);
    Map<String, Long> results =
        ... .collect
            (groupingBy
                (identity(),
                TreeMap::new,
                summingLong
                (String::length)));
    ...}
```

A constructor reference is used to create a `TreeMap`. See docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html
The collect() terminal operation typically returns a collection.

```java
void runCollectGroupingBy() {
    List<String> characters =
        Arrays.asList("horatio",
                      "laertes",
                      "Hamlet", ...);

    Map<String, Long> results = ...
        .collect
            (groupingBy
                (identity(),
                 TreeMap::new,
                 summingLong
                    (String::length)));

    ...
}
```

This "downstream collector" defines a summingLong() collector that’s applied to the results of the classifier function.

See [www.baeldung.com/java-groupingby-collector](http://www.baeldung.com/java-groupingby-collector)
The collect() terminal operation typically returns a collection.

```java
code
void runCollectReduce() {
    Map<String, Long>
    matchingCharactersMap =
    Pattern.compile("",")
    .splitAsStream
    ("horatio,Hamlet,...")
    ...
    .collect
    (groupingBy
    (identity(),
    TreeMap::new,
    summingLong
    (String::length)));
}
```

Convert a string into a stream via regular expression splitting!

See [docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#splitAsStream](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#splitAsStream)
A Stream Terminal Operation That Returns Collections

- The `collect()` terminal operation typically returns a collection.

```java
void runCollectReduce() {
  Map<String, Long>
  matchingCharactersMap =
  Pattern.compile("","").
  .splitAsStream
  ("horatio,Hamlet,...")
  .collect
  (groupingBy
   (identity(),
    TreeMap::new,
    summingLong
    (String::length)));
}
```

Collect the results into a `TreeMap` by grouping elements according to name (key) & name length (value)

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#groupingBy](docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#groupingBy)
End of Java Streams: the collect() Terminal Operation