

# The Java Streams collect() Terminal Operation (Part 3)

**Douglas C. Schmidt**

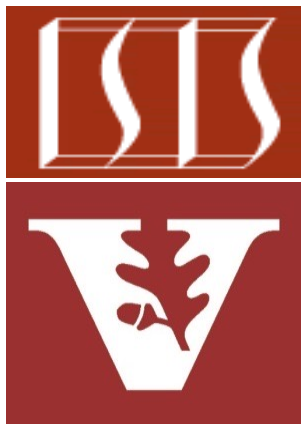
**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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()`
    - Know what a collector does
    - Recognize common Java pre-defined collectors & how to use them with `collect()`
  - Be aware of the Teeing Collector
    - Introduced in Java 12

## teeing

```
public static <T,R1,R2,R> Collector<T,?,R> teeing(Collector<? super T,?,R1> downstream1, Collector<? super T,?,R2> downstream2, BiFunction<? super R1,? super R2,R> merger)
```

Returns a Collector that is a composite of two downstream collectors. Every element passed to the resulting collector is processed by both downstream collectors, then their results are merged using the specified merge function into the final result.

The resulting collector functions do the following:

- **supplier**: creates a result container that contains result containers obtained by calling each collector's supplier
- **accumulator**: calls each collector's accumulator with its result container and the input element
- **combiner**: calls each collector's combiner with two result containers
- **finisher**: calls each collector's finisher with its result container, then calls the supplied merger and returns its result.

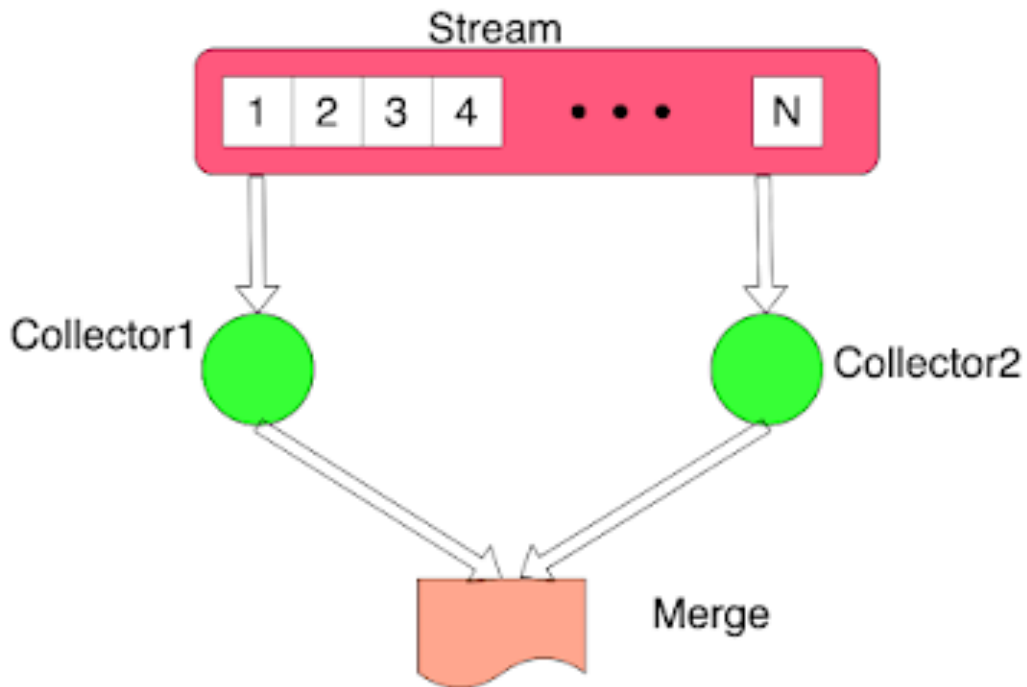
The resulting collector is `Collector.Characteristics.UNORDERED` if both downstream collectors are unordered and `Collector.Characteristics.CONCURRENT` if both downstream collectors are concurrent.

---

# Overview of the Teeing Collector

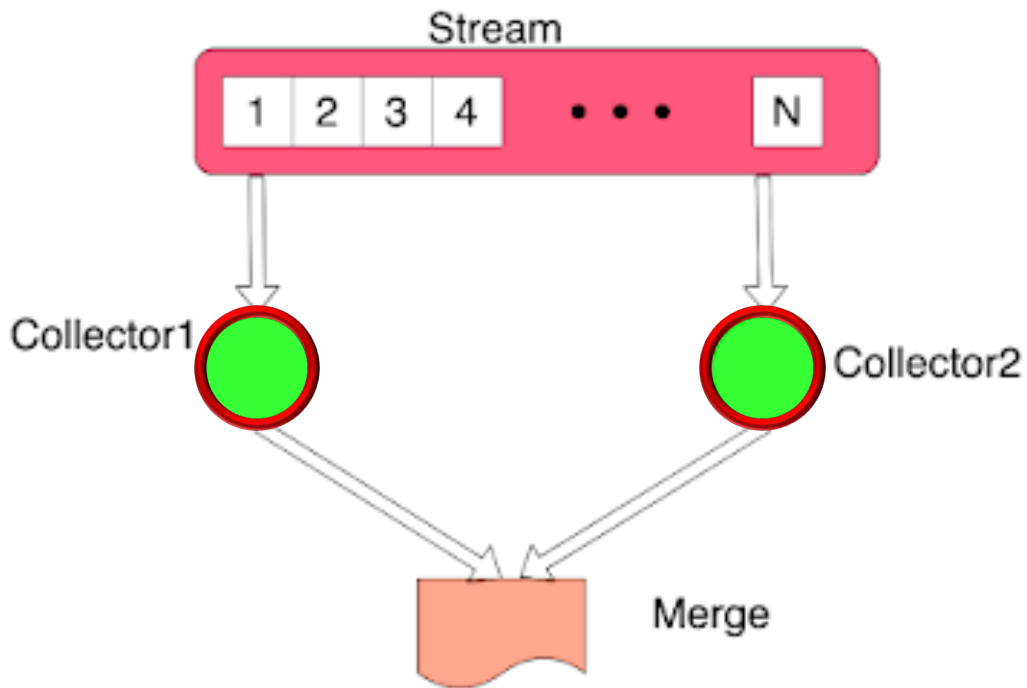
# Overview of the Teeing Collector

- The Teeing Collector returns a Collector that composes two downstream collectors



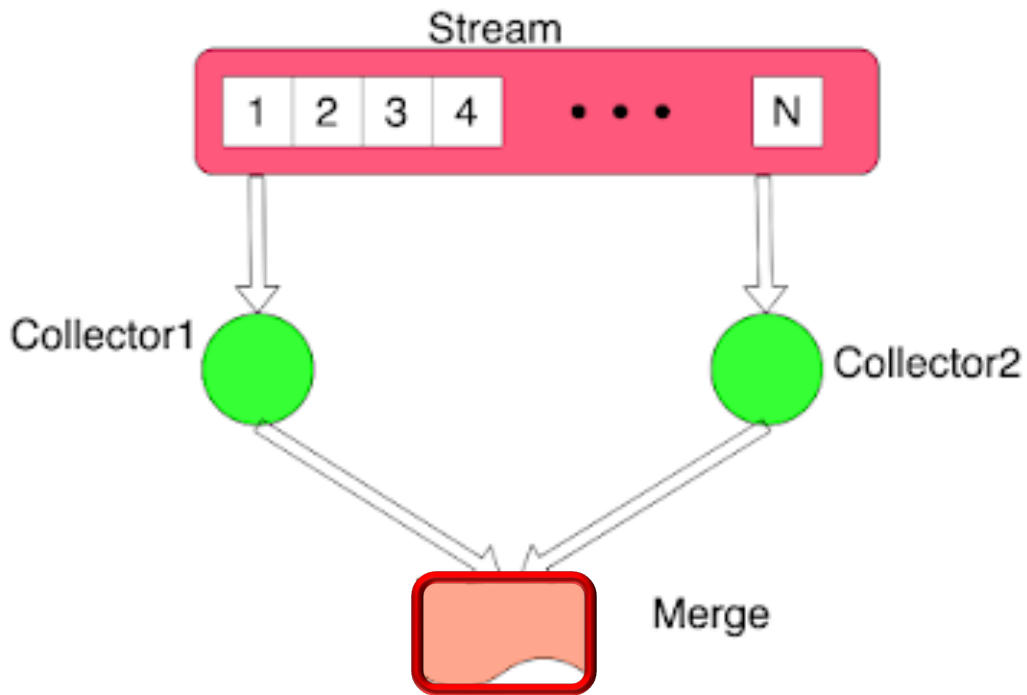
# Overview of the Teeing Collector

- The Teeing Collector returns a Collector that composes two downstream collectors
  - Every element passed to the resulting collector is processed by both downstream collectors



# Overview of the Teeing Collector

- The Teeing Collector returns a Collector that composes two downstream collectors
  - Every element passed to the resulting collector is processed by both downstream collectors
- Their results are then merged using the specified merge function into the final result



See [dzone.com/articles/java-12-the-teeing-collector](https://dzone.com/articles/java-12-the-teeing-collector)

---

# Applying the Teeing Collector

# Applying the Teeing Collector

---

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

```
void runTeeingCollector() {
    var results = sCharacters
        .stream()
        .map(Generators::capitalize)
        .sorted()
        .collect(teeing
            (filtering
                (startsWithHh(true),
                    toList()),
            filtering
                (startsWithHh(false),
                    toList()),
            Utils::concat);
}
```



# Applying the Teeing Collector

---

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

*Create a stream of all the Hamlet characters*

```
void runTeeingCollector() {  
    var results = sCharacters  
        .stream()  
        .map(Generators::capitalize)  
        .sorted()  
        .collect(teeing  
            (filtering  
                (startsWithHh(true),  
                    toList()),  
            filtering  
                (startsWithHh(false),  
                    toList()),  
            Utils::concat);  
}
```

# Applying the Teeing Collector

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

*Capitalize Hamlet characters consistently*

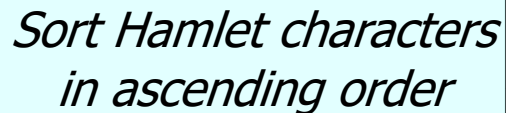
```
void runTeeingCollector() {
    var results = sCharacters
        .stream()
        .map(Generators::capitalize)
        .sorted()
        .collect(teeing
            (filtering
                (startsWithHh(true),
                    toList()),
            filtering
                (startsWithHh(false),
                    toList()),
            Utils::concat);
}
```

# Applying the Teeing Collector

---

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

*Sort Hamlet characters  
in ascending order*



```
void runTeeingCollector() {
    var results = sCharacters
        .stream()
        .map(Generators::capitalize)
        .sorted()
        .collect(teeing
            (filtering
                (startsWithHh(true),
                    toList()),
            filtering
                (startsWithHh(false),
                    toList()),
            Utils::concat);
}
```

# Applying the Teeing Collector

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

*Merge the results from two Collector operations on a Stream*

```
void runTeeingCollector() {
    var results = sCharacters
        .stream()
        .map(Generators::capitalize)
        .sorted()
        .collect(teeing
            (filtering
                (startsWithHh(true),
                    toList()),
            filtering
                (startsWithHh(false),
                    toList()),
            Utils::concat);
    ...
}
```

# Applying the Teeing Collector

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

*Collect a List of characters that start with 'H' or 'h'*

```
void runTeeingCollector() {
    var results = sCharacters
        .stream()
        .map(Generators::capitalize)
        .sorted()
        .collect(teeing
            (filtering
                (startsWithHh(true),
                    toList()),
                filtering
                    (startsWithHh(false),
                        toList()),
                Utils::concat);
    ...
}
```

# Applying the Teeing Collector

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

```
void runTeeingCollector() {  
    var results = sCharacters  
        .stream()  
        .map(Generators::capitalize)  
        .sorted()  
        .collect(teeing  
            (filtering  
                (startsWithHh(true),  
                    toList()),  
            filtering  
                (startsWithHh(false),  
                    toList()),  
            Utils::concat);  
    ...  
}
```

*Collect a List of characters  
that don't start with 'H' or 'h'*

# Applying the Teeing Collector

- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

```
void runTeeingCollector() {  
    var results = sCharacters  
        .stream()  
        .map(Generators::capitalize)  
        .sorted()  
        .collect(teeing  
            (filtering  
                (startsWithHh(true),  
                    toList()),  
            filtering  
                (startsWithHh(false),  
                    toList()),  
            Uutils::concat);  
}
```

*Merge the two lists together*

...

See [Java8/ex12/src/main/java/Utils/Utils.java](#)

# Applying the Teeing Collector

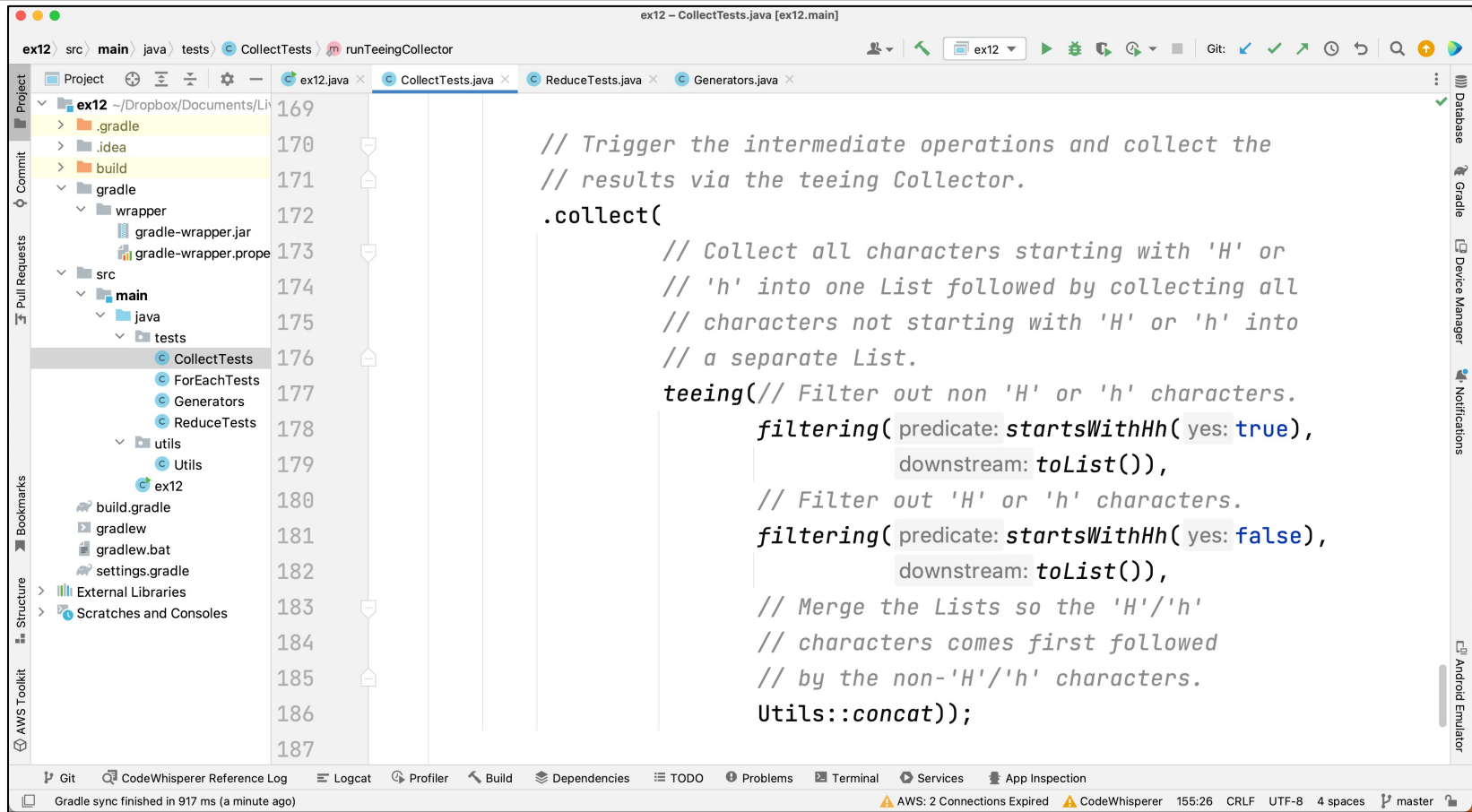
- Merge a stream so 'H'/'h' Hamlet characters comes first followed by the non-'H'/'h' characters

```
void runTeeingCollector() {  
    var results = sCharacters  
        .stream()  
        .map(Generators::capitalize)  
        .sorted()  
        .collect(teeing  
            (filtering  
                (startsWithHh(true),  
                    toList()),  
            filtering  
                (startsWithHh(false),  
                    toList()),  
            Utils::concat);  
    ...  
}
```

*Return the merged lists*



# Applying the Teeing Collector



The screenshot shows an IDE window titled "ex12 - CollectTests.java [ex12.main]". The left sidebar displays a project structure for "ex12" with a file tree. The main editor area shows the following Java code:

```
169
170 // Trigger the intermediate operations and collect the
171 // results via the teeing Collector.
172 .collect(
173     // Collect all characters starting with 'H' or
174     // 'h' into one List followed by collecting all
175     // characters not starting with 'H' or 'h' into
176     // a separate List.
177     teeing(// Filter out non 'H' or 'h' characters.
178         filtering(predicate: startsWithHh( yes: true),
179             downstream: toList()),
180         // Filter out 'H' or 'h' characters.
181         filtering(predicate: startsWithHh( yes: false),
182             downstream: toList()),
183         // Merge the Lists so the 'H'/'h'
184         // characters comes first followed
185         // by the non-'H'/'h' characters.
186         Utils::concat));
187
```

The IDE interface includes a top toolbar with icons for navigation and execution, a right sidebar with tool panels (Database, Gradle, Device Manager, Notifications, Android Emulator), and a bottom status bar with various tool icons and system information.

See [github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12](https://github.com/douglasraigschmidt/LiveLessons/tree/master/Java8/ex12)

---

# End of the Java Streams collect() Terminal Operation (Part 3)