Java streams utility methods for memoization

How to replay Java streams?

Unlike Iterable, where an execution pipeline can be executed as many times as we want, in a Java Stream we can iterate it only once.

Any call to a terminal operation closes the stream, rendering it unusable. Still, pre-caching items into a collection may be not viable for infinite streams.

Thus, streamemo library helps for items memoization without the need of recomputing them.

Usage

```
Random rnd = new Random();
Stream<Integer> nrs = Stream.generate(() -> rnd.nextInt(99));
Supplier<Stream<Integer>> nrsSrc = Replayer.replay(nrs);

// e.g. 88,18,78,75,98,68,15,14,25,54,22,
nrsSrc.get().limit(11).map(n -> n + ",").forEach(out::print);
out.println();

// Print the same previous numbers
nrsSrc.get().limit(11).map(n -> n + ",").forEach(out::print);
```

Note that you cannot achieve this result with an intermediate collection because `nrs` is an infinite stream. Thus trying to collect `nrs` incurs in an infinite loop. Only on-demand memoization like **`replay()`** achieves this approach.

Installation

First, in order to include it to your Maven project, simply add this dependency:

```
<dependency>
    <groupId>com.github.javasync</groupId>
    <artifactId>streamemo</artifactId>
    <version>1.0.1</version>
</dependency>
```

To add a dependency using Gradle:

```
dependencies {
  compile 'com.github.javasync:streamemo:1.0.0'
}
```

Changelog

1.0.1 (August, 2019)

Add the ability to close the original stream. Now the the `onClose()` method of a stream from the `Supplier.get()` will trigger a call to the original Stream's onClose() method. Contribution from shollander issue #2.

1.0.0 (June, 2018)

First release according to the article "How to Reuse Java Streams" published on DZone at Jun. 12, 18