Streams: Part 1

For all the exercises, start with a List of Strings similar to this:

- List<String> words = Arrays.asList("hi", "hello", ...);

1. Loop down the words and print each on a separate line, with two spaces in front of each word. Don't use map.

2. Repeat the previous problem, but without the two spaces in front. This is trivial if you use the same approach as in #1, so the point is to use a method reference here, as opposed to an explicit lambda in problem 1.

3. In the previous exercise, we produced transformed lists like this:

- List<String> excitingWords = StringUtils.transformedList(words, s -> s + "!");
- List<String> eyeWords = StringUtils.transformedList(words, s -> s.replace("i", "eye"));
- List<String> upperCaseWords = StringUtils.transformedList(words, String::toUpperCase);

Produce the same lists as above, but this time use streams and the builtin "map" method.

4. In the previous exercise, we produced filtered lists like this:

- List<String> shortWords = StringUtils.allMatches(words, s -> s.length() < 4);
- List<String> wordsWithB = StringUtils.allMatches(words, s -> s.contains("b"));
- List<String> evenLengthWords = StringUtils.allMatches(words, s -> (s.length() % 2) == 0);

Produce the same lists as above, but this time use "filter".

5. Turn the strings into uppercase, keep only the ones that are shorter than 4 characters, of what is remaining, keep only the ones that contain "E", and print the first result. Repeat the process, except checking for a "Q" instead of an "E". When checking for the "Q", try to avoid repeating all the code from when you checked for an "E".

6. The above example uses lazy evaluation, but it is not easy to see that it is doing so. Make a variation of the above example that proves that it is doing lazy evaluation. The simplest way is to track which entries are turned into upper case.

7. Take one of the previous examples where you produced a List, but this time output the final result as an array instead of a List. This is super-easy once you know how, and the class notes show this. But, the syntax looks *very* odd when you first see it.