The Stream Source Challenge

In the last video, I reviewed the following methods to generate a Stream.

Method	Finite	Infinite
`Collection.stream()`	X	
`Arrays.stream(T[])`	X	
`Stream.of(T...)`	X	
`Stream.iterate(T seed, UnaryOperator<T> f)`	X	X
`Stream.iterate(T seed, Predicate<? super T> p, UnaryOperator<T> f)`	X	
`Stream.generate(Supplier<? extends T> s)`		X
`**IntStream.range(int startInclusive, int endExclusive)`	X	
`**IntStream.rangeClosed(int startInclusive, int endExclusive)`	X	

In this coding challenge, you'll be using most of these methods to set up a few streams of your own.

The Stream Source Challenge

Generate the bingo ball labels as 5 different streams, using different Stream creation methods for each.

Assign each pipeline to a stream variable.

Concatenate the five streams together.

Apply the terminal operation, forEach, to the final concatenated stream, to print each label.

These should be printed in order as follows.

- B1-B15,
- I16-I30,
- N31-N45,
- G45-G60,
- O61-O75.

The Stream Source Challenge

Remember that the **map** intermediate operation, takes a Function, so you can return a different type, than the input stream element.

To do this, you'd use map to return a String, by executing a method or expression that takes an integer, and returns a string.

The generate method may be difficult to use, without creating side effects, or using other intermediate operations I haven't yet mentioned, but if you want a good challenge, you can play around with this one.

Go away and give that a try, and when you're ready, come back, and we can walk through my solution together.