**Streams**

**Always try to use a stream in place of a for loop when possible.  
Elements in a stream do not persist when the stream is finished.**

A Java lambda expression is a function that doesn’t belong to any class. It can be passed around as if it was an object.

**.filter( ) –** uses a lambda function ( **->** ) with a conditional check ( **! string.isEmpty( )** ) to create a new filtered list based on whatever the condition is. In this example, the filtered list will only contain elements that are not empty.

**.collect( ) –** Take the elements from a stream and store them in a concrete collection (new list).

**.reduce( ) –** Reduces a stream of elements into a single element. Useful for adding all numbers in a stream.

**.sorted( ) –** Sorts data in a stream into alphanumeric order. (0-9, A-Z).

**.map( ) –** Manipulate list items using a lambda function.

List<String> strings = Arrays.asList(“abc”, “”, “bc”, “efg”, “”);  
List<String> filtered = strings.stream( )**.filter**(string -> ! string.isEmpty( ) )**.collect**(Collectors.toList( ) );

List<Integer> myList = new ArrayList<Integer>();  
myList = {42, 6, 3, 44, 2};

myList.stream( )**.sorted( ).forEach**(System.out::println);

myList.stream( )**.filter(**element -> element %2 == 0**).collect(**Collectors.toList);

myList.stream( )**.map**(num -> num \* 10)**.forEach**(System.out::println);