Java Programming
Lab Streams 2

Intermediate Methods

1. Lab objectives

The purpose of this lab is to get you familiar with both the stream pipeline methods and the process designing a pipeline.

2. Lab Setup

You can do this whole lab in one project with different packages for the different parts

3. Lab Solutions

Solutions for each lab are provided in the repository in the labs directory for each module. These are not the only possible solutions and your solution to the labs may be a different in some ways than the reference solutions.

Since the demos are based on the labs, if you are having problems getting started, refer to the demo code, also in repository, to give you some inspiration.

Part 1: Map, Filter and Peek

1. For this part of the lab, generate an unsorted `List<Integer>` of about 20 integers random between 1 and 10. For example the lab solution uses

```
public static void main(String[] args) {
    List<Integer> data = Arrays.asList(
            8, 9, 4, 3, 1,
            9, 1, 2, 9, 8,
            5, 3, 4, 10, 6,
            5, 7, 5, 1, 6);
```

2. Write a stream that implements the following steps
    1. Use the operator `peek(x -> System.out.println("First peek " + x))` to display the elements of the stream as they are processed.
    2. Filter out all elements that are less that five.
    3. Use the operator `peek(x -> System.out.println("Second peek " + x))` to display the elements of the stream after the filter.
    4. Replace any occurrence of 9 with a 0.
    5. Sort the result.
    6. Return the result as a list of integers using `Collect(Collectors.toList())`
3. Print the result to ensure that the code executed correctly
4. Notice that it when the stream executed, the peek operation showed that the stream was processed sequentially at each step from the first element to the last, but processing on an element is terminated as soon as it is filtered out of the stream
5. Move the filter operator to just after the sorting step and put the second peek just before the filter
    1. Run the code and see how it processes the streams
    2. What conclusions can you come to about ordering filters with respect to the other steps?

Part Two: More Intermediate Methods

1. You can modify the code from the previous part to do this part.
2. Write a stream that removes the first 10 elements using `skip(10)` and print the results
3. Try skipping more elements than are in the stream (`skip(100)`) and see what happens
4. Write a stream that process only the first 10 elements in the stream using `limit(10)`
5. Write a stream that removes duplicates using the `distinct()` operator

Note that both `distinct()` and `sorted()` have to work with all of the elements in the step. This is an example of functio// Lazy Invocation demonal programming. We are telling the stream that we want to remove duplicates of sort the stream, but we do not have to supply code to describe how to do these operations.

Part Three: FlatMap

1. Create a list of lists of countries and print it out to ensure you understand the structure of the data

```
List<String> asia = Arrays.asList("Japan","Korea", "China", "Laos");
List<String> europe = Arrays.asList("England","France", "Sweden", "Laos");
List<String> sa = Arrays.asList("Brazil","Argentia", "Peru", "Guyana");
List<String> fifa =  Arrays.asList("Brazil","Argentia", "England", "France");
List<String> nato =  Arrays.asList("Sweden", "England", "France");


List< List<String> > countries = new ArrayList< List<String> >();
countries.add(europe);
countries.add(asia);
countries.add(sa);
countries.add(fifa);
countries.add(nato);

System.out.println(countries);
```

2. Use flatMap() to flatten this into a single stream of strings.
3. The flatMap() operation will have to create a stream of strings from each of the list of string

```
List<String> nations =
        countries.stream()
        .flatMap(x -> x.stream())
        .collect(Collectors.toList());

System.out.println(nations);
```

4. Print out the result to ensure that it is a single list of strings.
5. Sort the result into alphabetical order and remove any duplicates

End of Lab