Assignment 2

Tools & Techniques for Large-Scale Data Analytics (CT5105)
NUI Galway, Academic year 2019/2020, Semester 1

- ***Submission deadline (strict): Wednesday, 16th October, 23:59.*** *Late submission only with a medical or counsellor's certificate.*
- *Put all source code files (and other answers, if any) into a single .zip archive with name "YourName_Assignment2.zip" and submit via Blackboard by the deadline.*
- *Include all source code files (that is, files with name ending .java) required to compile and run your code, including all sources for your classes, interfaces, etc.*
- *Unless specified otherwise in the question, use only plain Java for this assignment (no external libraries).*
- *All submissions will be checked for plagiarism.*
- ***Use Java comments to explain your source code. Missing or insufficient comments lead to mark deductions.***

Question 1 [40 marks]

Suppose you want to analyze the data produced by a couple of weather stations (meteorological stations). Create a class `WeatherStation` with three attributes (fields): the *`city`* where the station is located, the station's *`measurements`* (a list of objects of class `Measurement`), and a *static* field *`stations`* (a list of all existing weather stations – you need this list only for the next question). Also, create a class `Measurement`. Objects of class `Measurement` should have attributes *`time`* (an integer, representing the time of the measurement) and *`temperature`* (a double number).

Add a method `maxTemperature(startTime, endTime)` to this class which returns the maximum temperature measured by the weather station between `startTime` and `endTime`.
Implement this method using Java 8 Stream operations, as far as possible. Also, add a `main`-method where you call your method using some test data (of your own choice) and print the result.

Question 2 [60 marks]

Add a method `countTemperatures(t1,t2,r)` to your class `WeatherStation` from the previous question. The method should return a list which contains two pairs: 1) temperature t1 paired with the number of times a temperature in the interval `[t1-r..t1+r]` has been measured so far by any of the weather stations in *`stations`*, and 2) temperature t2 paired with the number of times a temperature in the interval `[t2-r..t2+r]` has been measured so far by any of the weather stations in *`stations`*. Note that a single measurement contributes to both counts in case it is within both intervals.

Example: if there are two weather stations in list *`stations`* and the first station has measured temperatures 20.0, 11.7, -5.4, 18.7, 20.9 and the second station's measurements are 8.4, 19.2, 7.2, then `countTemperatures(19.0,10.8,2.1)` should return the list ((19.0,4), (10.8,1)) (because there are in total (i.e., considering all stations) 4 measurements in the range between 19.0-2.1 and 19.0+2.1 and 1 measurement in the range from 10.8-2.1 to 10.8+2.1).

PTO

For computing the result, you need to use an “emulated” MapReduce approach. That is, your code should resemble the MapReduce approach but using only Java >=8 (without any cluster framework like Spark or Hadoop). Also, you need to make use of Java 8 Streams (as far as possible) and parallel stream processing (where appropriate).

Important: This question doesn’t ask for the fastest or shortest Java code! Instead, your solution should reflect, as far as possible, the MapReduce phases “Map” and “Reduce” as described in the lecture, and it should appropriately use (key,value)-pairs in each phase (so you will also need some data structure for representing pairs, e.g., a class with two fields). There should also be some kind of grouping of values (or (key,value)-pairs) by their keys, resembling the “Shuffle” step between “Map” and “Reduce” (see lecture notes). Additional operations, e.g., for filtering, can be added if needed.

Hints: Studying the “word count” example from the lecture will be helpful (although there are some differences) - e.g., weather stations might correspond to inputKeys, certain temperatures might correspond to words… Also, take a look at the available methods in interface Stream (https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html), e.g., flatMap for “flattening” nested streams…

Finally, add code to your `main`-method which calls your `countTemperatures` method using a few test stations and some test measurement data, and prints the result.