CSCI 340
Data Structures and Algorithms
Spring 2019

Project 2 – Median of an Integer Stream

Due date: Thursday, February 28
Points: 25

In some companies it is common to ask candidates to code during an interview. For example **both Microsoft and Google ask candidates to code and simultaneously talk their way through how they would solve a problem.** On the idiot side of these sorts of questions are problems like FizzBuzz. On the more complex side are problems like this one.

Background

Recall from basic statistics that the median is a measure of central tendency (the middle of the data). For an odd length set of data it is the middle element of the equivalent sorted array. The median of an even length set of data is the average of the middle two elements (in the equivalent sorted array). For example the median of: 2, 4, 7 (or 4, 7, 2) is 4. And the median of 2, 4, 7, 8 (or 7, 8, 2, 4) is (4+7)/2 = 5.5. The median is the same, regardless of the order of the data. If the data is relatively normal the median will have a value very close to the mean. If the data contains extreme values, however, the median may be a preferred statistic to the mean, as the mean can be unduly influenced by the extreme values.

In computing, a ***stream*** is a sequence of potentially infinite data, where the data is processed one element at a time. Streams can take data from all sorts of sources. For example, in Java `System.in` is an input stream and `System.out` is an output stream. Similarly, if we have a sensor (thermometer, hygrometer, barometer, or whatever) connected to a computer, that sensor will generate a stream of data.

When we have a stream of input data, we often want to calculate running statistics as we process the stream. For example, if the stream is of temperatures in a furnace smelting iron ore, we may want to keep track of the maximum and minimum temperature encountered so far, and want to know the median temperature so far. All of these would tell us something about the quality of the smelting process.

Your Assignment

You are to write a class named `StreamMedian`. The class should have a constructor and two methods:

```
public void insert(Integer i)
```

This method should add the next element of the stream to the data used for calculating the median.

`public double getMedian()`
This method should find and return the median of all the data inserted.

The solution you are to implement guarantees optimal runtimes. We will to use two priority queues. For our purposes here, I will call them `bigger` and `smaller`.
`bigger` will be a min heap containing the larger half of the data encountered so far.
`smaller` will be a max heap containing the smaller half of the data encountered so far. Note to make a max heap, you will need to implement a Comparator.

Constraints

1. If there are an even number of data, `smaller` and `bigger` should be of the same size.
2. If there are an odd number of data we allow one of the priority queues to be one larger. For ease of programming, let’s say that smaller is the one we allow to have the additional element.

Note that we have two constraints (the smaller-larger constraint and the size constraint). You need to keep shifting data between the two priority queues to make sure the constraints are maintained.

Calculating the median is simple:
If the total number of elements in both PriorityQueues is even, we average the maximum from `smaller` and the minimum from `bigger`.
If the total number of elements is odd, the median is the largest element in `smaller`.
Note that we do NOT remove any data from the priority queues. We look at the max and min, but leave them in the queue.
Since there is only a simple if statement, `getMedian()` is `O(1)`.

Hint:
This is a very short assignment. If your code, with comments, runs more than 100 lines total, you are doing something wrong.

What to turn in:
Submit a single java source file named `StreamMedian.java`. You do not need to submit the test main your instructor has provided. He will test your code against a different test main. As always, do not use packages.