CS 204, Program #1: Simulation in an Array

Due Wednesday, Sept. 9th by 11:55 p.m.

Summary: A program to simulate the spread of a heat in a rod.

The Problem: In Physics, there are many kinds of forces (heat, electrical, magnetic, gravitational) where there can be more than one force acting on an object of interest. A common phenomenon is that the force at one "cell" is the average of the forces in adjacent "cells". We want to examine what happens when we put fixed forces at two points along a rod, and then let the system come to equilibrium. For example, imagine that we have a rod, which we will consider divided into 6 segments. We apply a temperature of 80º at one end of the rod, a temperature of 50º at the other end, and assume all other temperatures are "room temperature" in Celsius. Storing the 6 segments in an array, the array at the beginning of our simulation will be:

80 | 20 | 20 | 20 | 20 | 50

We now begin the simulation. After one step, the first and last segments will have the same temperatures (we're keeping them at those temperatures), and each other cell will have a temperature equal to the average of the temperature in it and its neighboring cells *from the previous step.* Thus the new temperatures will be:

80 | 40 | 20 | 20 | 30 | 50

At step 2, the process repeats, and the temperatures will be:

80 | 46.67 | 26.67 | 23.33 | 33.33 | 50

And so on. If we let this run forever, we should reach a stable equilibrium with temperatures of

80 | 74 | 68 | 62 | 56 | 50

At this stage, each temperature other than the two fixed endpoints has a temperature exactly equal to the average of the two adjacent values. That's what we mean by a "stable equilibrium".

Programming Requirements:

1) Your package must be named "heatingRod", and your only class named "HeatingRod".
2) You must have three class variables titled rodLength, leftTemperature, and rightTemperature. (rodLength is int; the others are floats), appropriately defined.
3) You must have a method titled "getInput", which takes no input and returns one int. See the "Sample Run" for the details of the input formatting. It should ask for the values of the three class variables and store them there. It should also ask for the number of iterations to use for the simulation, and that should be the return value.
4) You must have a method titled "initializeRod" which creates the array of floats, sets the left and right temperatures, and sets all other temperatures to 20º (which is the European standard for "room temperature", in Celsius of course). It should return that array.
5) The 20º of requirement #4 should be a program constant, properly defined. (For example, "room temperature" in the US is standardized to 25º Celsius.)
6) You must have a method titled "updateTemps", which calculates the new temperatures in the rod after *one* iteration of the heat dispersion. The new temperature in cell *i* (so long as *i* is not one of the endpoints of the rod) should be the average of three values: the old temperature of cell *i*, and the temperatures of its two neighbors (cells *i*–1 and *i*+1). This method should take as input the array of the current rod temperatures and return the array of the new rod temperatures.
7) You must have a method titled "simulateHeating" which takes as input the initial temperature array *and* the number of iterations the user requested in requirement #3. The two parameters must be listed in that order. The method should return the final temperatures.

You must have a method title “toString” which takes an array of temperatures and converts it to a string, in the format shown in the “Sample Run”. To make sure your numbers are converted to one decimal place accuracy, use a line similar to:

```
String s = new DecimalFormat ("#.0").format( temperatureValue );
```

8) Your main method must organize these other methods to accomplish the program goal.
9) No method should be more than 9 lines long. If any method is too long, figure out how to use private helper functions to reduce the length of that method.
10) Your code must satisfy all of the provided JUnit tests (to be distributed soon).
11) Your output must meet all the formatting requirements of the Sample Runs. Common mistakes are to miss the space between the computer output and your input, and to have your input appear on a separate line.

Sample Run #1:

```
What is the size of the array? 6
Temperature at the left: 80
Temperature at the right: 50
How many iterations should we run? 1
Final temperatures, to one decimal place:
80.0, 40.0, 20.0, 20.0, 30.0, 50.0
```

Sample Run #2:

```
What is the size of the array? 6
Temperature at the left: 80
Temperature at the right: 50
How many iterations should we run? 2
Final temperatures, to one decimal place:
80.0, 46.7, 26.7, 23.3, 33.3, 50.0
```

Sample Run #3:

```
What is the size of the array? 6
Temperature at the left: 80
Temperature at the right: 50
How many iterations should we run? 52
Final temperatures, to one decimal place:
80.0, 74.0, 68.0, 62.0, 56.0, 50.0

Note (which is not part of the sample run: This is where this scenario first
comes to an apparent stable equilibrium, to within round-off error.
```

Sample Run #4:

```
What is the size of the array? 20
Temperature at the left: 80
Temperature at the right: 50
How many iterations should we run? 52
Final temperatures, to one decimal place:
80.0, 72.0, 64.2, 56.9, 50.3, 44.4, 39.4, 35.4, 32.4, 30.4, 29.2, 29.0, 29.6,
30.9, 33.0, 35.6, 38.7, 42.3, 46.1, 50.0
```