COP 2551 – Introduction to OOP
Program #5 Due: Friday, 22 April, noon

Please note: There is no 'late turnin.'

Note: This is an easy program; worth 80 points. You will only need a Main class with one additional static method (see below). Its main purpose is to acquaint you with 2d arrays, and that's it.

Using NetBeans 6.9.1 or later version you are to write a Java program using OOP principles to accommodate the following functionality

Using the input file *Countries.txt* (which contains a number of European countries, their capitals, and the population of these capitals in addition to region number), you are to build a 4 x 12 two-dimensional array of integers (one row per region) from these capital populations, average the numbers in each row and display the array as well as each row's average. Each record in the input file has two fields of interest to you necessary to build your two-dimensional array: the region number and the population. (You will need a substring to extract those two numbers, as you have done in previous programs.)

Task 1: Build the two-Dimensional Arrays.
Build your arrays such that each row contains a list of integers (population of capital cities) for 'that' region. Each array may well have a different number of non-zero entries. Unused entries must be filled with zeroes. Specifically, row 1 would have six non-zero columns: 480000 720000 500000 100000 1360000 and 15000 namely arrayName (0,0), (0,1), (0,2), etc. that is, all the populations from countries that were in region 1. Fill remaining entries with zeros for that row. Your second row will have all the populations from those countries that are in region 2, etc. Since there are four regions, your two-dimensional array will have four rows. The number of non-zero entries in each row depends on the number of countries in that region.

Task 2: Print out the Two-Dimensional Array.
You may use the routines in your book discussed at length in class. These are very close to exactly what you will need. See comments below on printing out headers to make this look nice. Use a nicely formatted header above the array that articulates what you are displaying.

Task 3: Average the populations per row.
Write a **method** that expects you to pass it a **single** row of the array, like the parameter: arrayName[i] and returns the average population of those entries in that row (or equivalently in that Region. Call the method four times, once for each row in your array. Each time you call this method, pass the average population back to main each time as an integer.

Task 4: Compute the average population of each region
When a method that is passed an array that returns the average of the numbers (population) in that array. This should be a static method in the Main class.

Task 5: Overall Displaying

Here you are to print headers (report and column) as shown below:

European Country Capital Population (By Region Number)

Region Number	Region	Capital Averages
1	Northern Europe	n,nnn,nnn
2	Western Europe	n,nnn,nnn

etc.

Overall Average Population Per Capital is: n,nnn,nnn

Grade Sheet for Program 5
COP 2551
Spring 2011

Name: _____

Source Code – 20 points **Points earned:** _____
Indentation
Internal comments
Scope terminators
Overall program structure

Javadoc – 10 points **Points earned:** _____
Appropriateness and completeness of comments
See link on my web page. Follow the procedure. Be sure to include Javadoc comments in your source code before you generate the Javadoc.
Each method must be preceded with Javadoc statements and these need to be visible in the generated Javadoc .html files.

Detailed Design – 10 points **Points earned:** _____
Pseudo-code. See many examples on my web page and previous emails.

Outputs – 40 points **Points earned:** _____
Accuracy and Format; Are they correct
Skip lines in between displayed numbers for readability. Make the output look professional!
Include headers / descriptors as you may feel appropriate.

Please start early. Note the due date. You have plenty of time if you start right away.

Program must run totally and correctly to receive a passing grade. Do a little (one step) at a time!