CSE114 Fall 2023 : Assignment 5

Due: Oct 13, 2023 at 11:59 PM [KST]

Read This Right Away

For the due date of this assignment, don't go by the due date that you see on Brightspace unless you have set your timezone to KST. By default, Brightspace shows times in EST/EDT in the US.

Directions

- At the top of every file you submit, include the following information in a comment
  - Your first and last name
  - Your Stony Brook email address
- Please read carefully and follow the directions exactly for each problem.
- Your files, Java classes, and Java methods must be named as stated in the directions (including capitalization). Work that does not meet the specifications may not be graded.
- Your source code is expected to compile and run without errors. Source code that does not compile will have a heavy deduction (i.e. at least 50% off).
- You should create your program using a text editor (e.g. emacs, vim, atom, notepad++, etc).
- Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.

What Java Features to Use

For this assignment, you are *not* allowed to use more advanced features in Java than what we have studied ***at the time I post the assignment to Brightspace***. If you have a question about features you may use, please ask!

What to Submit

Combine all your .java files from the problems below into a single zip file named as indicated in the **Submission Instructions** section at the end of this assignment. Upload this file to **Brightspace** as described in that section.

Multiple submissions are allowed before the due date. Only the last submission will be graded.

Please do **not** submit `.class` files or any that I did not ask for.

Partial vs. Complete Solutions

Your programs should compile and run without errors, both compile-time and run-time errors! Please do not submit programs that do *not* even compile! Its better to submit a partial solution that compiles and runs as opposed to an almost complete solution that does not even compile. A program that does not compile even if it is very close to being perfect would only receive less than 50% maximum of the possible score for the program! This policy applies not only to this problem set but also to all the other ones in the remainder of the semester. I will not repeat this in each problem set though. So, please remember this.

Naming Conventions In Java And Programming Style In General

Please use these conventions as you establish your *programming style.* Programming professionals use these, too.

- **Names:** Choose informative names,
  - e.g. `hourlyRate` rather than `hr` for a variable name.
  - `CheckingAccount` rather than `CA` for a Java class name.
- **Class names:** We start a class name with an uppercase letter as I have in my examples: `Hello` not `hello` or `HELLO`. For multi-word names you should capitalize the first letter of each word, This is called 'Pascal' case.
  - e.g. `UndergraduateStudent`, `GraduateStudent`, `CheckingAccount`
- **Variable names:** We start a variable name with a lowercase letter. This is called 'Camel' case.
  - e.g. `count`, `hourlyRate`, averageMonthlyCost
- **Method names:** Naming convention for method names is the same as that for variable names.
- **Use of white space:** Adopt a good indentation style using white spaces and be consistent throughout to make your program more readable. Most text editors (like emacs and vim) should do the indentation automatically for you. If this is not the case, see me and I can help you configure your setup.

I will not repeat these directions in each assignment, but you should follow this style throughout the semester.

Problem 1 (10 points)

Create a class named **RowsAndColumns** in the file **RowsAndColumns.java**.

The first problem involves writing two methods. One will sum each row of a 2 dimensional array. It will return a single dimension array of ints holding the sums from each row. The prototype is:

```
public static int[] sumRows(int[][] inArray)
```

The next method you will write is one that will take a 2 dimensional array and sum each column. The prototype is:

```
public static int[] sumColumns(int[][] inArray)
```

Write a main to test both of these methods. As one test, use the following 2D array as input:

```
int[][] inputMatrix = {{10, 15, -1, 22, 11},
        {100, 200, 150, -30, 27},
        {11, 44, 2, 1, -15},
        {44, 89, 10, 21, -2},
        {44, 55, 105, 205, 305}};
```

You should create a couple of additional 2 dimensional arrays of different sizes to verify your code works regardless of size (for instance, create an array that is not square like the above one so it will have a different number of rows than it has columns).

Write some code in your main that prints the results returned on one entry per line.
With the above array as input, the results should look like the following:

```
Row Sums:
57
447
43
162
714
Column Sums:
209
403
266
219
326
```

Problem 2 (15 points)

Now, in the same java file, add methods that will average the Rows and average the columns in a 2D array. Important note: These methods will return a 1 dimnesional array of doubles since they are averages of several values.

The prototypes are:

```
public static double[] averageRows(int[][] inArray)

public static double[] averageColumns(int[][] inArray)
```

Add code in your main that calls these new methods and prints the results returned on one entry per line. Note you should use printf() in main when printing the returned arrays and limit the values to 2 decimal places.

Using the same input array as shown in problem 1, the results should look like the following:

```
Row Averages:
    11.40
    89.40
     8.60
    32.40
   142.80
Column Averages:
    41.80
    80.60
    53.20
    43.80
    65.20
```

Problem 3 (10 Points) Extra Credit!

This is an optional problem worth 10 points (above the standard 25 point assignment)!

For the above 4 problems, adapt your code to handle 'jagged arrays'. This is not difficult for summing and averaging the rows. However, it will be a challenge to complete this for summing and averaging columns!

Here is a sample input array on which you can test your code:

```
    int[][] inputMatrix2 = {{10, 15, -1},
                    {100, 200, 150, -30, 27, 59, -201, 3},
                    {11, 44, 2, 1, -15},
                    {44, 89, 10, 21, -2, 10},
                    {44, 55, 105, 205, 305}};
```

Indicate in your submission if you tried implementing this code and include another set of test output that uses a jagged array like inputMatrix2 above..

Submission Instructions

Please follow this procedure for submission:

1. Place the deliverable source files (`RowsAndColumns.java`) into a folder by themselves. The folder's name should be CSE114_PS5_<yourname>_<yourid>. So if your name is Alice Kim and your id is 12345678, the folder should be named 'CSE114_PS5_AliceKim_12345678.

2. Compress the folder and submit the zip file.
    a. On windows, do this by pressing the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brightspace dropbox for **Assignment5**
    b. On mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Brightspace dropbox for **Assignment5**.
3. Navigate to the course brightspace site. Click **Assignments** in the top navigation bar.
    a. Click **Assignment5** in the content area. Under **Submit Assignment**,
    b. Click **Add a File** (left button).
    c. In the dialog box that appears, select **My Computer.** Drag the .zip file into the Upload area of the dialog box (or click the **Upload** button and select from a file selection dialog box).
    d. Click the **Add** button.
    e. You may add comments in the text box below.
    f. Click the **Submit** button 🡐 ***Very important! If you forget to do this, we cannot access your assignment and it will not be graded!***

.