ASSIGNMENT 2

COMP-202, Summer 2017, All Sections

Due: Sunday, May 27^{th} 2017, (23:59)

Please read the entire PDF before starting. You must do this assignment individually.

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

To get full marks, you must:

- Follow all directions below
- Make sure that your code compiles
  - Non-compiling code will receive a very low mark
- Write your name and student ID as a comment in all .java files you hand in

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Create a file called `Counting.java`, and in this file, declare a class called `Counting`. This class should ask the user when the computer should stop counting.

```
When should I stop counting to?
10 // User typed this
I am counting until 10: 1 2 3 4 5 6 7 8 9 10
```

Warm-up Question 2 (0 points)

For this question you have to generalize the last question. The user will give you the number they want the computer to count up to and the step size by which it will do so.

```
When should I stop counting to?
25 // User typed this
What step size should I use?
3  // User typed this
I am counting to 25 with a step size of 3:
1 4 7 10 13 16 19 21 24
```

Warm-up Question 3 (0 points)

This program prints the outline of a square made up of # signs. It should ask the user for the length of the sides in number of #s. Using two loops, you should be able to draw the outline of a square as follows.

```
How big of a square?
10     //User typed this
##########
#        #
#        #
#        #
#        #
#        #
#        #
#        #
#        #
##########
```

N.B. It is normal that the square does not appear to be a perfect square on screen as the width and the length of the characters are not equal.

How would you extend your program to output a rectangle with width and length specified by the user?

Warm-up Question 4 (0 points)

Let's write a method incrementally:

1. Start by writing a method called `getRandomNumber` that takes no inputs, and returns a random `double` between 0 (included) and 10 (excluded).
2. Now, modify it so that it returns a random `int` between 0 and 10.
3. Finally, let the method take two integers `min` and `max` as inputs, and return a random integer greater than or equal to `min` and less then `max`.

Part 2

The question in this part of the assignment will be graded.

Question 1: Matrix Calculator (100 points)

You are to write your very first interactive java program!. You will write a Matrix Calculator which will perform basic operations with matrices. In particular, the program allows users to:

1. Fill two matrices by user-input.
2. Fill two matrices by random numbers.
3. Print Matrices.
4. Check if the matrices are magic.
5. Transpose the matrices.
6. Sum two matrices.
7. Multiply two matrices.
8. Exit the program.

Now that we now what actions our Matrix Calculator should perform, let's see which methods we need to make it work. All the code for this question must be placed in a file named MatrixCalculator.java (already provided to you in the course web-page).

0 Setting up the main method

We have already set up the main method for you. This method consist of a `switch` block that will model the possible execution paths based on the option entered by the user[1]. In other words, this `switch` block is in charge of calling the appropriated methods based on the user input (i.e., one of the eight possible options listed above). This `switch` block is located inside a `do while` loop that will keep showing to the user the main menu until he/she decides to exit the program.

1 Fill two matrices by user-input

The method called `FillUser` takes one integer as input and returns a (int[][]) matrix filled with the data provided by the user. The input value correspond to the size of the matrix (i.e., the number of rows and columns). Please note that our Matrix Calculator will operate only with square matrices (i.e., a matrix with the same number of rows and columns). You can assume that the user will enter only valid data (i.e., integer values). You must write the code needed to perform the filling of the matrix inside the method called `FillUser`.

2 Fill two matrices by random numbers

The method called `FillRandom` takes one integer as input and returns a (int[][]) matrix filled with random numbers produced by the computer. The input value correspond to the size of the matrix (i.e., the number of rows and columns). Please note that our Matrix Calculator will operate only with square matrices (i.e., a matrix with the same number of rows and columns). Note that it is ok to have repeated integer numbers in your matrix. You must write the code need to perform the filling of the matrix inside the method called `FillRandom`. To fill in the data of the matrix, you will have to generate random numbers between 1 and the size of the matrix (both included). If you have any doubts on how to achieve

[1] You can get more information about switch statements in the following link: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

this, make sure you first understand the warm-up Question 4, where you are asked to build a method called `getRandomNumbers`.

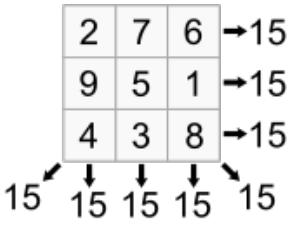
3 Print matrices

The method called `PrintMatrix` takes one (int[][]) matrix as input and print in the screen the matrix (i.e., print all the content of the matrix). You need to implement your code in the void method called `PrintMatrix`.

4 Check if a matrix is magic

The method called `IsMagic` takes one (int[][]) matrix as input and returns a boolean value (i.e., `true or false`) acknowledging if the input matrix is magic or not. In a magic matrix, every row, column, and diagonal add up to the same number. The following figure (taken from wikipedia) shows and example of a 3×3 matrix for which every row, column and diagonal add 15.

Figure 1: A magic matrix.



5 Transpose a matrix

The method called `Transpose` takes one (int[][]) matrix as input and returns a (int [][]) matrix in which the matrix have been flipped over its diagonal. In other words, this method switches the row and column indices of the matrix (i.e., the first row of the input matrix is now the first column of the returned matrix, the second row of the input matrix is now the second column of the returned matrix, and so on). The below example transpose a matrix of order 3×3.

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \longrightarrow A^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

6 Sum two matrices

The method called `Addition` takes two (int[][]) matrices as input and returns a (int [][]) matrix representing the sum of the two input matrices. The addition must be performed entry-wise as shown in the following example.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} + \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix} = \begin{bmatrix} a+j & b+k & c+l \\ d+m & e+n & f+o \\ g+p & h+q & i+r \end{bmatrix}$$

7 Multiply two matrices

The method called `Multiply` takes two (int[][]) matrices as input and returns a (int [][]) matrix representing the multiplication of the two input matrices. if X is an $n \times n$ matrix and Y is an $n \times n$ matrix, their matrix product XY is an $n \times n$ matrix, in which the n entries across a row of X are multiplied with the n entries down a column of Y and summed to produce an entry of AB.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nn} \end{bmatrix} \times \begin{bmatrix} y_{11} & y_{12} & y_{13} & \dots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \dots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & y_{n3} & \dots & y_{nn} \end{bmatrix} = \begin{bmatrix} (XY)_{11} & (XY)_{12} & (XY)_{13} & \dots & (XY)_{1n} \\ (XY)_{21} & (XY)_{22} & (XY)_{23} & \dots & (XY)_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (XY)_{n1} & (XY)_{n2} & (XY)_{n3} & \dots & (XY)_{nn} \end{bmatrix}$$

where $(XY)_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$

What To Submit

You have to submit the file called `MatrixCalculator.java` to myCourses - Assignment 2. You can also submit (optionally) a file called `Confession.txt`. In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise they would not.

Marking Scheme

Up to 30% can be removed for bad indentation of your code as well as omitting comments, coding structure, or missing files. Marks will be removed as well if the class and methods names are not respected.

User Options

Option 1:	10	points
Option 2:	10	points
Option 3:	10	points
Option 4:	15	points
Option 5:	15	points
Option 6:	15	points
Option 7:	25	points
	100	**points**