CSC220 Assignment 2
Object Oriented Programming

The goal of this week's assignment is:

1. Start working with the concept of class and object
2. Learn to implement methods for your class

Things you must do:

1. There are many details in this assignment. Make sure you read the whole thing carefully before writing any code and closely follow this instruction.
2. You must complete your assignment individually.
3. Always remember Java is case sensitive.
4. Your file names, class names, and package name must match exactly as they are specified here.
5. Your project must include the methods you implemented in the lab.

Things you must not do:

1. You must not change the file names, class names, package names.
2. You must not change the signature of any of these methods (name, parameters, …). Just fill in the missing code inside them.
3. You must not create a method with a new name.
4. You must not create any different class.

You should have created a new project during the lab and you will be using the same project for this assignment. You are going to continue working on that project and complete the partial implementation of the class that represents 2D matrices. Matrices are used in many fields of mathematics and computer science, including computer graphics and physical simulations. There is a matrix arithmetic primer at the end if you need it.

Your job is to complete several methods defined that are incomplete. Since this Matrix class will represent a Matrix object, its methods are not static. That is, in order to call these methods, you must call them from an object of the Matrix class. You should have already implemented `toString` method in the lab, I would recommend to start implementing the equal method, as it is easier and will help you verify that your other methods are working correctly.

I advise you to read the "Matrix Arithmetic Primer" at the end of this document first before starting the assignment.

Important Note: You are encouraged to write more tests in your main function to make sure you have implemented all methods properly. However, submit your code with the original MatrixTester.java provided. We will first run the main function provided to make sure your code is working with the examples we provided and then, we will test your code with new and different examples.

Part 0

- Grab the new version of **MatrixTester.java** from the assignment ZIP files and replace the version you used in the lab. Make sure you have all methods from the lab working properly!

Part 1 – `equals`

- input: an object to compare to
- output: a boolean indicating whether the other Object represents the same Matrix as this one

- notes:
    - Part of this method is done for you, i.e. determining whether the other Object is indeed a Matrix.
    - You will write the rest of this method. If the other Object is a Matrix, you must determine if the two matrices have the same dimensions and values, in the same order. If so, return true, otherwise return false.

Part 2 – `transpose`

- input: none
- output: A new matrix that is the transpose of this matrix

Part 3 – `add`

- input: the Matrix to be added (right-hand side of addition)
- output: a new Matrix that is the result of this Matrix added to the input Matrix
- notes:
    - This function must make sure that the two matrices being added are the same size and return null if they aren't.

Part 4 – `mult`

- input: the Matrix to be multiplied by (the right-hand side of a multiply)
- output: a new Matrix that is the result of this Matrix multiplied by the input Matrix
- notes:

- This function must make sure that the two matrices being multiplied have compatible dimensions (number of rows and columns) for matrix multiplication, and return null if they aren't. See the primer section below on Matrix multiplication.

- This function must automatically determine the size of the new Matrix (which may not be the same as either of the original matrices). The size of the new matrix is the number of rows of the left matrix, by the number of columns in the right matrix. See the matrix multiplication primer section below.

Remarks

- Make sure to submit your assignment by uploading your **Lab02** folder into your **csc220-cXXXX** folder by the deadline **(Thursday @ 11:59pm)**
- **For all your assignments, please start early and seek help early (either from the instructor or the TAs).**

Matrix Arithmetic Primer

Matrix multiplication: First of all, not all pairs of two matrices can be multiplied together -- they must have *compatible dimensions*. Specifically, the matrix on the left must have the same number of columns as the number of rows in the matrix on the right.

For example, a 2x3 matrix can be multiplied by a 3x4 matrix, since the number of columns in the first matrix (3) is equal to the number of rows in the second.

[2x3] * [3x4] -> valid, result is [2x4]

Matrix multiplication is not commutative, which means reversing the order of the arguments may not always work:

[3x4] * [2x3] -> invalid

Make sure you pay attention to which Matrix is on the left and which is on the right. In this method, we assume that the argument to the times function is the one on the right, and the Matrix calling the method is the one on the left, for example:

```
m1.mult(m2) -> m1 is on the left, and m2 is on the right (m1 * m2)
```

The dimensions of the resulting matrix are the number of rows in the left matrix by the number of columns in the right matrix, as seen from the valid example above.

Once you have determined if the matrices are compatible for multiplication, computing the actual result is as follows:

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}\times\begin{bmatrix}7 & 8\\9 & 10\\11 & 12\end{bmatrix}=\begin{bmatrix}(1*7+2*9+3*11) & (1*8+2*10+3*12)\\(4*7+5*9+6*11) & (4*8+5*10+6*12)\end{bmatrix}$$

or

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}\times\begin{bmatrix}7 & 8\\9 & 10\\11 & 12\end{bmatrix}=\begin{bmatrix}58 & 64\\139 & 154\end{bmatrix}$$

Specifically, move across a row of the left matrix, and down a column of the right matrix, multiplying each element together, and summing them all up. This resulting value fills one space of the result matrix, in the [i][j] position, where [i] is the row traversed from the left matrix, and [j] is the column traversed from the right matrix.

Matrix addition: Matrix addition is a bit simpler than multiplication. Two matrices can be added together only if they have the exact same size, that is, the left matrix must have the same number of rows as the right matrix, and the left matrix must have the same number of columns as the right matrix. To compute the resulting matrix, simply add the two elements from the left and right matrices together in the corresponding position. In other words,

```
result[i][j]= left[i][j] + right[i][j], for every i, j:
```

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}+\begin{bmatrix}7 & 8 & 9\\10 & 11 & 12\end{bmatrix}=\begin{bmatrix}8 & 10 & 12\\14 & 16 & 18\end{bmatrix}$$