Assignment 1

This assignment is due Thursday, January 17 at 5:00pm. You must work alone on this assignment.

There are a lot of details in this assignment. Make sure you read the whole thing carefully before writing any code!

This assignment will give you a chance to brush up on your basic Java programming skills learned in CS 1410 that you may not have used in a while. You must complete this assignment individually (no groups).

This assignment is not representative of the average difficulty of assignments in this class. It should be fairly easy if you are coming fresh out of CS 1410. The rest of the assignments in this class will be more difficult.

2D Matrices

In this assignment you will complete a partial implementation of a class that represents 2D mathematical matrices. Matrices are used in many fields of mathematics and computer science, including computer graphics and physical simulations.

The class has been started for you, but you must fill in several of the methods. Start by downloading Matrix.java and MatrixTester.java. These files contain the definition of a class called "Matrix", that has the necessary members and constructors for a matrix, and a class with a main method to test the Matrix class. Create a new package in Eclipse called "assignment1", and paste these two files in to that package. After downloading them, you can just drag and drop them on to the assignment1 package in Eclipse, when it asks whether to copy or link to the files, select copy.

Your file names, class names, and package name must match exactly as they are specified here.

These files have several methods defined that are incomplete. Your job is to complete these methods. 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.

Make sure you download and become familiar with the two provided files before reading any further. Start with the `toString` and `equals` methods, as they are easier and will help you verify that your other methods are working correctly.

Don't change the signature of any of these methods, just fill in the missing code inside them.

`times`

- 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 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 section below.

`plus`

- 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.

`toString`

- input: none
- output: a String representing this Matrix
- notes:
    - The format of the String must be exactly as specified for grading purposes. The String should contain each element in the Matrix from left to right, top to bottom. Since this Matrix class is represented as a 2D array (called `data` in

the class provided), that means that the item at `data[0][0]` is the top-left item, `data[0][numColumns-1]` is the top-right item, `data[numRows-1][0]` is the bottom-left item, and `data[numRows-1][numColumns-1]` is the bottom-right item. This is called row-major order. For example, the test Matrix created in the MatrixTest.java file provided creates a Matrix `m1` with the 2D `int` array `{{1, 2, 3},{4, 5, 6},{7, 8, 9}}`. The return value from calling `m1.toString()` should be the following String:

```
1 2 3
4 5 6
7 8 9
```

There is a single space character after each number (including the last number of each row: there is a space after the 3, 6, and 9), and a single newline ("\n") at the end of each row.

`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, that determines whether or not the other `Object` is a `Matrix` or not.
    - 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`.

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, however, so 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.times(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:

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

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.

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

Testing

Part of your grade for every assignment in this class will come from testing. You must add tests similar to the example tests in the `main` function in `MatrixTester`. You must convince the TAs and I that you have sufficiently tested your code. This includes writing tests that will exercise your error-checking code, for example, trying to multiply two matrices with incompatible dimensions. You should test all of your functions and all error cases, as well as valid cases. Use your `equals` and `toString` methods to verify the results of your tests, as seen in the example test in the file given to you. You should therefore implement and test your `equals` and `toString` methods before moving on to `times` and `plus`.

Note that we will not test empty matrices. Assume all matrices will have dimensions at least 1x1

Handing in

Use the button at the bottom of this page to hand in your Matrix.java and MatrixTester.java

We need both of your files for credit!

Available from:	Thursday, 10 January 2013, 5:25 AM
Due date:	Thursday, 17 January 2013, 5:05 PM

Submission feedback

Kendal Gifford
Monday, 21 January 2013, 2:23 PM

Grade: 94.00 / 100.00

```
---Automated Grading Results for Assignment 1---

toString:    15/15

equals:       15/15

Failed 3x2 * 2x3: expected "6 17 18; 7 16 21; 3 7 9" but got "5 12; 13 26"
Failed 2x2 * 2x3: threw java.lang.ArrayIndexOutOfBoundsException: 2
times:        24/30

plus:       30/30

Subtotal:    84/90
Testing:    5/5
Style:       5/5
Total:       94/100
```

Submission

- Matrix.java
- MatrixTester.java

No further submissions are allowed.