CSC220 Lab02

Object Oriented Programming

The goal of today's lab (and this week's assignment) is:

1. Start working with the concept of class and object
2. Learn about how to implement and work with (multiple) constructors
3. Implement a toString method for our class
4. Implement various methods for your class

Important Note: There are many details in this lab. Make sure you read the whole thing carefully before writing any code and closely follow this instruction. Consult the lab instructor if you ran into difficulty. **Don't change the signature of any of these methods, just fill in the missing code inside them.**

Part 0 - Create a project for this lab in Eclipse

1. You are going to create a new project for this (and each of the remaining labs). You learned how to create a project in Eclipse in the previous lab. Create a new Java project and call it **Lab02** (with no space and no other name – notice the capital 'L')
2. Create a package inside your project and call it **lab02** (no space and no other name – all lowercase).
3. Log into Blackboard and grab the two Java files from the Lab02 zip file (google drive link) and place them in the"src" folder of the project you just created.
4. Refresh your Eclipse to make sure you are looking at the template code.

Part 1 – 2D Matrices

1. Do not modify anything in MatrixTester just yet.
2. Open Matrix.java and you will see three constructors: default constructor, constructor 1, and constructor 2.
3. Leave the default constructor as is.
4. Implement constructor 1
5. Implement toString method (skip the constructor2 for now).

   note: 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},{2, 5, 6}}`. The return value from calling `m1.toString()` should be the following String:

```
1 2 3
2 5 6
```

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

6. Go back to MatrixTester and make sure your implementation is correct by printing the sample matrix (M1 – Experiment1) provided. Your output should be **exactly** as follows:

```
M1 is:
0 0 0
0 0 0
done.
```

7. If your output looks different **do not** change the print line or MatrixTester. Go back to Matrix.java and debug your class implementation.
8. Obviously, if you have any compile/run-time error, there might be something wrong with your constructor and/or your toString method. I highly recommend to try to debug it by yourself before seeking help.
9. After having the toString and constructor 1 working, we want you to do a little experiment to make sure you understand yesterday's lecture. That is, how multiple constructors and default constructor work.

Experiment 1.1

10. In this step, comment out all the constructors and save both files and make sure you understand the error showing up in MatrixTester.java. You should be able to explain why we have that error (if not, consult the class slides – constructor section – if you are still in doubt consult with your lab instructor).
11. Now, in MatrixTester.java, comment out the definition of M1 right after the line that says "Experiment 1" and uncomment the M1 definition line after "Experiment 2". Do not modify Matrix.java in this step (i.e., all constructors are commented out). You should not get any compile errors. Why? – consult the lecture notes for help and/or the lab instructor.

    Note: if you still get an error in this step you are doing something wrong – go back, double check your previous steps.
12. Now try to run the code, you'll see that there is no run-time error either. Why? –

    consult the lecture notes for help and/or the lab instructor.

Experiment 1.2

1. Go back to Matrix.java and uncomment constructor 1 that you implemented. Make sure the default constructor and constructor 2 are both commented out.

2. Make sure you have the definition of M1 corresponding to Experiment 2 uncommented and the definition of M1 corresponding to Experiment 1 commented out.
3. Why do you get a compilation error? – consult the lecture notes for help and/or the lab instructor.
4. uncomment your default constructor and make sure you understand why the error in MatrixTester.java goes away.

Experiment 1.3

1. You are using the M1 definition for Experiment 1 and 2. Comment out both definitions and uncomment the M1 definition for Experiment 3 (you are supposed to uncomment three lines in this case). Don't worry about the errors showing up, they'll go away as soon as you have done the next step.
2. Go back to Matrix.java and implement constructor 2.
3. Run MatrixTester.java and your output should **exactly** look like the following

```
M1 is:
1 2 3
2 5 6
done.
```

4. If your output looks any different continue debugging Matrix.java (make no changes in the main function).
5. After you successfully implemented constructor 2. try your code with the other sample matrix we provided M2 and you should see the following output:

```
M2 is:
4 5
3 2
1 1
done.
```

6. You are done with the lab!

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.

Submission Guidelines

You should have already created your csc220-xxxxxxx folder in your university box account and know how to submit your assignment through BOX (you can consult the tutorials in blackboard again if you have forgotten).

Inside your csc220-xxxxxxx folder in box, you should only have a single folder called **Lab01** after submission of Lab01 and Assignment01.

In order to submit Lab02, log into your university box account, go into your csc220-xxxxxxx folder and upload **Lab02** folder from your Eclipse workspace.

Now you should see two separate folders in your csc220-xxxxxxx folder (in BOX):

one should be called **Lab01** and the other should be **Lab02.**

Remember, you will **re-upload your Lab02 folder** after you are done with the assignment (it is **OK** if Box complains that “some uploads have failed”; if you look closely at what uploaded, you should see a checkmark next to your JAVA files :-)

Feel free to upload your code to Box when you have completed the lab, but remember that both the “lab” portion and the “assignment” portion of the assignment are due:

Thursday night @ 11:59pm!