Introduction to Computer Science, IDC Herzliya, 2016

HW 4: Arrays, Part II

In this assignment you will practice working with two-dimensional arrays.

Start by reading the entire assignment thoroughly and understanding all the requirements. Before implementing any code, we recommend writing pseudo-code *on paper*, tracing its execution *on paper*, and convincing yourself that you got the logic right.

Matrix Operations

This project focuses on creating a Java class that supports matrix operations. We provide the skeleton of a class named `MatrixOps`, and your assignment is to complete the missing code in that class. As we started doing in lecture 5.1, we will use two-dimensional arrays to represent matrices. We now turn to describe the operations that you will have to implement. The implementation details will be described separately, and later in the document.

1. **Equality**

   Two matrices A and B are said to be equal if (i) they have the same dimensions, and (ii) for every element in both matrices, $A(i,j) = B(i,j)$.

2. **Transpose**

   The *transpose* of a matrix A, denoted A^T, is the matrix whose rows equal the columns of A. Specifically, for every element in both matrices, $A^T(i,j) = A(j,i)$. For example, consider the following matrix:

   ```
   1 2
   3 4
   5 6
   ```

   The transpose of this matrix is:

   ```
   1 3 5
   2 4 6
   ```

3. **Symmetry**

   A *square* matrix is a matrix in which the number of rows equals the number of columns. A square matrix A is said to be *symmetric* if $A(i,j) = A(j,i)$ for every i and j. Here is an example of a symmetric matrix:

   ```
   1 1 8 9
   1 0 4 8
   8 4 2 7
   9 8 7 2
   ```

4. **Identity**

   The *identity* matrix is a square matrix that has 1 in every element along its main diagonal, and 0 elsewhere. Here is an example of an identity matrix of size 4:

   ```
   1 0 0 0
   0 1 0 0
   0 0 1 0
   0 0 0 1
   ```

5. **Product**

   Suppose that matrix A has $n1$ rows and m columns, and matrix B has m rows and $n2$ columns. The product $C = A \cdot B$ is a matrix of $n1$ rows and $n2$ columns, in which every element is computed as follows: $C(i,j) = \sum_{k=1}^{n} A_{i,k} B_{k,j}$

   For example if A is

   ```
   1  0 -2
   0  3 -1
   ```

   and B is

   ```
    0  3
   -2 -1
    0  4
   ```

   then their product is

   ```
    0 -5
   -6 -7
   ```

6. **Subset**

   Given a matrix A and two coordinates specified by the four integers $i1, j1, i2, j2$, the subset of A is the sub-matrix whose top-left element is $A(i1, j1)$, and whose bottom-right element is $A(i2, j2)$. Assume that $i2 \geq i1$ and $j2 \geq j1$, and that all these indexes are less than or equal to the relevant matrix dimensions.

   For example, consider the matrix:

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

   The subset of the matrix defined by the four integers 2, 2, 3, 4 is:

   ```
   4 2 7
   1 3 5
   ```

7. **More**

   The supplied `MatrixOps` class skeleton includes two more matrix-oriented functions: `print`, for printing a given matrix, and `random`, for generating a matrix with random integer values. These functions are self-explanatory.

Testing

In this homework we expect you to write your own testing code, and submit it along with your work. For each one of the implemented class methods, your testing code must demonstrate that the method operates correctly. Make sure that you test extreme cases (but note that you don't have to test for invalid indexes -- see comment below). Don't over-test: if the method passes a certain test case, there is not need to test other similar cases. Put your tests in the `main` method. Note that this method already contains some testing code. Feel free to modify or delete this code, as you see fit. If you want, you can use private methods to make your testing code more readable and organized. Use your judgment.

As it turns out, the product of a matrix and its transpose is a symmetric matrix. Also, if A is a square matrix of size m, and I is the identity matrix of size m, then $A \cdot I = A$, and $I \cdot A = A$. Part of your testing should be code that demonstrates these two theorems, using random matrices (one test for each theorem).

Implementation notes

1. Start by implementing the `print` method. This method is essential for debugging all the other methods.
2. Note that in mathematics, the indexes of a matrix start from 1. For example, the top-left element of the matrix A is denoted $A(1,1)$. However, in Java, the same element is denoted `A[0][0]` (if you use two-dimensional arrays to represent matrices). We assume that users of the `MatrixOps` class will use mathematical notation (indexes starting from 1). For example, if the user wants to see the value of A(3,4), the code should supply `A[2][3]`. In many cases (e.g. when you have to operate on the entire matrix) this issue is irrelevant. In other cases it can cause confusion. So, handle it.
3. Throughout this project, there is the danger of operating on illegal indexes. In reality, your code must guard against this possibility. However, for the purpose of this homework, you should assume that all the given indexes are always valid. That is, the indexes are non-negative integers which are less than or equal to the relevant matrix dimensions.
4. Make sure to document your code. All additional functions should be documented using JavaDoc.
5. Altogether, you have to implement 8 methods. The signatures of 6 of these methods are given in the `MatrixOps` class, and "all" you have to do it write their implementation. For the method that generates a random matrix and the method that generates a subset of a given matrix, you also have to design the method signatures.

Submission: Before submitting any program, take some time to inspect your code, and make sure that it is written according to our **Java Coding Style Guidelines**. Also, make sure that each program starts with the program header described in the **Homework Submission Guidelines**. Any deviations from these guidelines will result in a deduction of points.

Submit the following file only:

- `MatrixOps.java`

Deadline: Submit your assignment no later than Sunday, December 18, 23:55.