Computer Science 3A

Practical Assignment 1

16 February 2023

Time: 16 February 2023 — 17h00 Marks: 100

Practical assignments must be uploaded to `eve.uj.ac.za` **before** 17h00. Late submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student. You **must** include your Javadoc for each question in your submission. If you do not include it, **you will get a zero for your entire question**.

Operations on Matrices are very common operation in many areas of Computer Science, include AI, Computer Graphics and Cryptography. In this practical you are required to implement a program to demonstrate various Matrix operations. You must make use of Java as the language of implementation, and your program should consist of various classes where you implement your solution.

You are required to implement the following matrix operations:

- Matrix-Matrix Addition
- Matrix-Matrix Multiplication
- Matrix-Scalar Multiplication
- Matrix-Scalar Addition
- Matrix Transpose

Consider two Matrices A and B, where both are $m \times n$:

$$A_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \quad B_{m,n} = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,n} \end{bmatrix}$$

Matrix-Matrix Addition

$$A + B = A_{i,j} + B_{i,j} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} & \cdots & a_{1,n} + b_{1,n} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} & \cdots & a_{2,n} + b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} + b_{m,1} & a_{m,2} + b_{m,2} & \cdots & a_{m,n} + b_{m,n} \end{bmatrix}$$

Matrix-Scalar Addition

Consider a constant c that must be added to the matrix A.

$$c + A = c + A_{i,j} \begin{bmatrix} c + a_{1,1} & c + a_{1,2} & \cdots & c + a_{1,n} \\ c + a_{2,1} & c + a_{2,2} & \cdots & c + a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c + a_{m,1} & c + a_{m,2} & \cdots & c + a_{m,n} \end{bmatrix}$$

Matrix-Scalar Multiplication

Consider a constant c that must be added to the matrix A.

$$cA = cA_{i,j} \begin{bmatrix} c \times a_{1,1} & c \times a_{1,2} & \cdots & c \times a_{1,n} \\ c \times a_{2,1} & c \times a_{2,2} & \cdots & c \times a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c \times a_{m,1} & c \times a_{m,2} & \cdots & c \times a_{m,n} \end{bmatrix}$$

Matrix Transpose

Consider the Transpose of the Matrix A:

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \cdots & a_{m,n} \end{bmatrix}$$

Matrix Multiplication

To multiply two matrices together involves a bit more calculation. If you multiply an $m \times n$ matrix by an $n \times p$ matrix, the result will be an $m \times p$ matrix. Technically multiplying the elements in the resulting matrix are calculated by taking the dot product of the row of the first matrix with the column of the second matrix.

The dot-product for two $1 \times n$ matrices is calculated as follows:

$$P = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}, Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}$$
$$P \cdot Q = p_1 q_1 + p_2 q_2 + \cdots + p_n q_n$$

So each element in two matrix that are multiplied together can be indicated as follows:

$$[AB]_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + \cdots + A_{i,r}B_{r,j} = \sum_{r=1}^{n} A_{i,r}B_{r,j}$$

Consider the following example:

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix} = \begin{bmatrix} ag + bj & ah + bk & ai + bl \\ cg + dj & ch + dk & ci + dl \\ eg + fj & eh + fk & ei + fl \end{bmatrix}$$

You are required to implement a Java Program that realises the above operations. You must implement the following items:

`Matrix` — a class that contains a 2-dimensional array of values that represents a Matrix.

1. Your matrix should use a 2-dimensional array of values to hold the underlying data.
2. Methods to access elements, rows and columns.
3. Methods to obtain information about the Matrix, such as the dimensions.
4. Appropriate constructors.
5. The matrix operations listed above.
6. Your class should make use of Generics, if your class only works with one type you will lose 20% of the total mark.
7. Your class must implement the `IMatrix` interface.
8. Your class should implement all of the indicated Exceptions in the appropriate functions.

A Java interface has been provided to you that you must use to implement your class. All methods in this interface must be implemented completely by your class. Your application must be runnable from an executable JAR file. You must submit your JAR file and all source files for your application.

You must provide the appropriate Javadoc documentation for every function, class, and instance variable you create. The Javadoc documentation for your assignment should be generated and uploaded as a separate zip file on EVE.

The following files must be submitted to EVE:

1. *studentnumber*_p1.zip

Creating Arrays with Generics

Creating an array in a generic object can be difficult. It is possible to do it with Reflection in Java 7, but this is incompatible with other versions of Java and can be problematic to get the base class type from passed generic parameters.

The easiest way is to work with a generic array is look at the following class:

```
public class MyArray<E> {
    private E[] arr;

    public MyArray(Integer s) {
        genArray(s);
    }

    public void genArray(int s) {
        final Object[] a = (E[]) new Object[s];
        arr = (E[])a;
    }

    public void setElement(int x, E value) {
        arr[x] = value;
    }

    public E getElement(int x) {
        return arr[x];
    }
}
```

This does not produce code that is very safe, but it does create a generic array. You can run into type erasure problems if the objects are not handled correctly. It is easily extendible for 2-dimensional arrays. You should not use the above class in your solution, just the concepts presented in the above class.

Marksheet

1. Matrix: default and overloaded constructors [7]
2. Matrix: numberRows and numberCols [2]
3. Matrix: getRow [5]
4. Matrix: getCol [5]
5. Matrix: accessor and mutator for element [5]
6. Matrix: addMatrix [7]
7. Matrix: addScalar for both Double and Integer [12]
8. Matrix: multiplyScalar for both Double and Integer [12]
9. Matrix: multiplyMatrix [12]
10. Matrix: transpose [4]
11. Matrix: toString [4]
12. Main:command line or Gui with random matrix generation [10]
13. Compilation and Correct execution [15]