COSC 20203 Fall 2018 Lab Assignments

For each lab you will have to provide a zipped file with the following required contents

1. A **one page** report describing
   a) the problem
   b) how it was solved
   c) the most interesting classes you developed

2. A Zipped file with all the complete **.java** files required to run your application
3. For Lab2 and on the zipped file will have to include a clickable jar that
   Will run the lab.

4. For Lab 2 on the zipped file will have to incude Javadoc folder containing the documentation of the html files of your classes. Note that in order to document it properly you must use documentation guidelines and you must save each class in a unique file with name of the class.

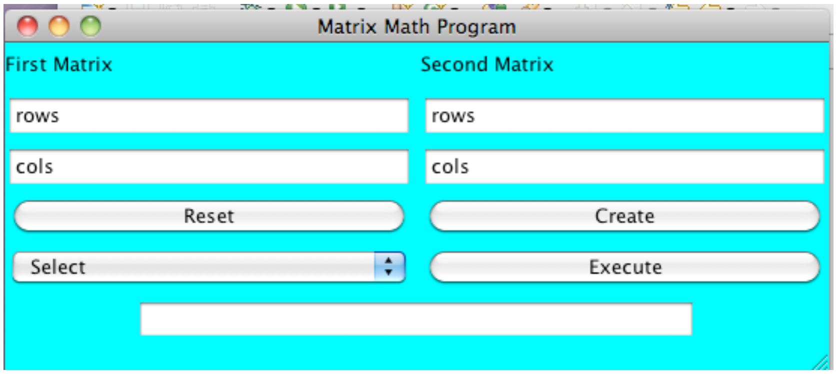
5. Two pages of screenshots of your program running

6. Use **D2L** to submit your zipped file with the complete Lab Assignment

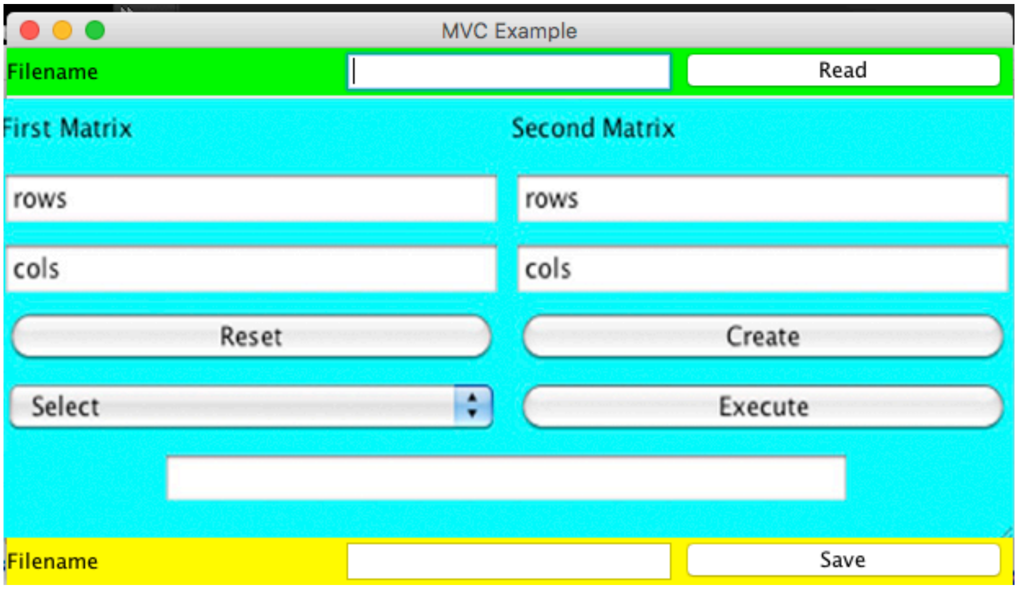
Lab #1 **CoSc 20203** **Fall 2018**

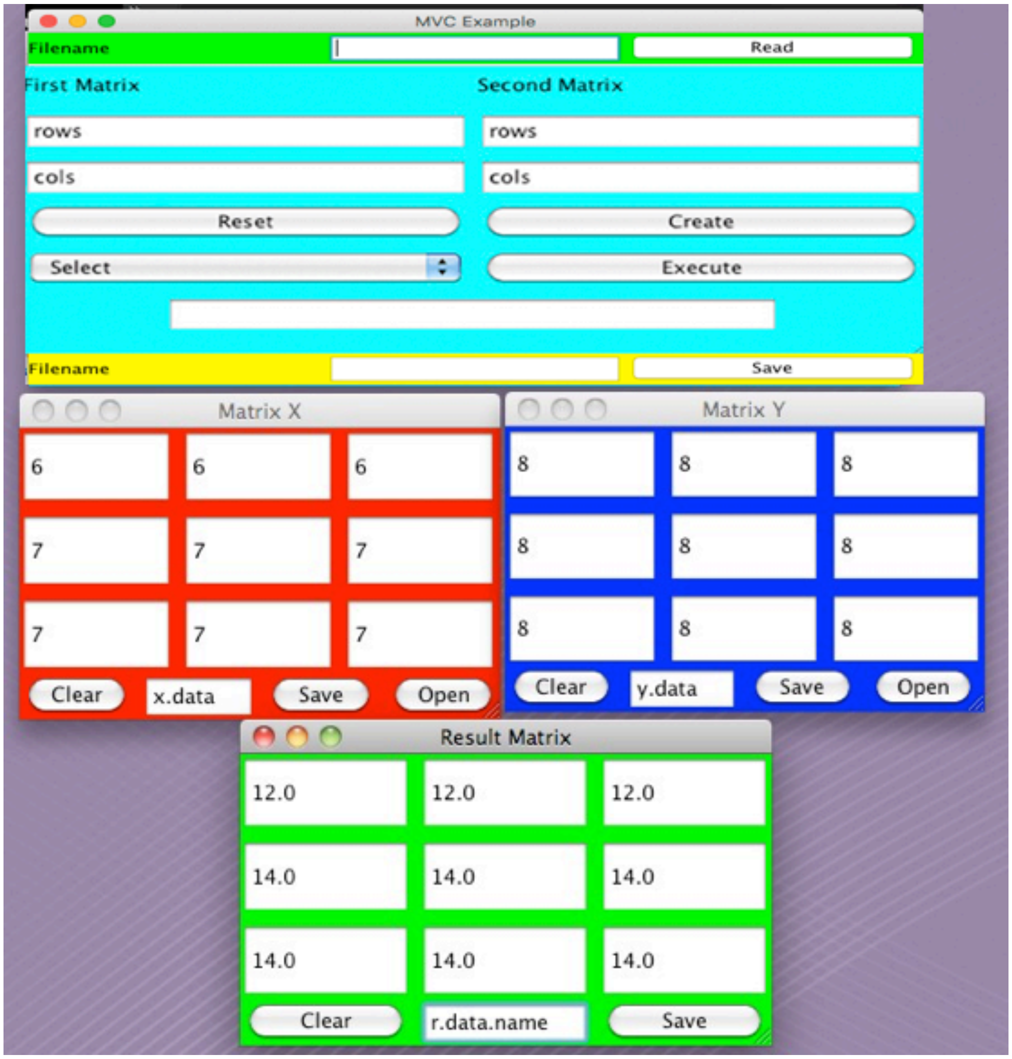
Due: Thursday Sep 6 Midnight

Using the MVC concepts write a set of Java classes to perform matrix operations (such as Addition, Subtraction, Transpose, Multiplication).
Your program should include a basic GUI interface using swing widgets to select the various options of your program. Here is a possible example of the GUI, but you can modify it according to your own design.



The output of your program should consist of as many frames as necessary to display the values of the initial matrix as well as the resulting matrix. **Your program should include the capacity to save the results as objects in external files using object persistence as discussed in class.** Therefore the view should include the means to select the file names, perhaps something like this





Matrix X

6	6	6
7	7	7
7	7	7

x.data

Matrix Y

8	8	8
8	8	8
8	8	8

y.data

Result Matrix

12.0	12.0	12.0
14.0	14.0	14.0
14.0	14.0	14.0

r.data.name

In Brief

1) You should provide a nice **GUI(VIEW),** a good **interaction** (**CONTROL**) and a **good translation scheme**(**MODEL**) with both binary and hexadecimal display. Each component should be in a separate class to provide a good **MVC** design
2) Data input must be validated to avoid exceptions, i.e. using try{}/ catch() {} instructions
3) In matrix multiplication, compatibility of the matrices must be performed. i.e. the number of columns of the first matrix must be equal to the number of rows of the second matrix
4) **JTextFields [ ] [ ]** must be used to display and read the data and display the result
5) **double [ ] [ ]** arrays must be used to store the data
6) Provide good documentantation