Exam Project
Computer Programming / Introduction to Programming
June 1st, 2020

Details about how to submit the project

Deliver your project as a zip file named summer-exam-2020-USER-STUDENTID.zip (substitute 'USER' with your name and 'STUDENTID' with your student id number). **This zip file must contain two files (another zip file and a jar file).**

The **first** file is another zip file obtained by using the **export** command in Eclipse. This can be done by first choosing the File>Export menu command. Then select as destination in the 'General' folder the **Archive File** type. Be sure to select the right project, the zip format and check the radio button "Create directory structure for files" (default). Include in the zip file all the files (bin, sources, resources), the .classpath and .project files. You can consult this page https://mcuoneclipse.com/2012/05/07/exporting-and-importing-projects-in-eclipse/ for an example. I must be able to **import** your Eclipse project, compile and execute the program **without any additional library** that is not already included in the standard Java API.

The **second** file is an executable JAR file containing your application. It is obtained from Eclipse by first choosing File>Export menu command. Then select the 'Java' folder and inside it 'Runnable JAR file', and then click on 'Next' button. In the 'Launch configuration' you must select the class that contains the main of your app (pay attention, there should be many classes). Then in the "Export destination' field enter a path and file name with extension .jar where you want to save your jar file. Below in the 'Library handling' radio button select 'Package required libraries into generated JAR'. Do not click 'Save as ANT script'. Then press the button 'Finish'.

Double check that one of your colleagues, by using an OS different from yours, can successfully execute your project (jar file) without any problem. **Do not put in your code absolute references to files**. **The project should be compiled with Java 10.**
Note that **if I cannot run your project or the application does not find the required files** (see the requirements) you will **fail the exam**.

Upload this zip file in OLE before the deadline: **June 15th 2020 at noon.**

Student Code Ethics

Students are expected to maintain the highest standards of academic integrity. Work that is not of the student's own creation will receive no credit. Remember that you cannot give or receive unauthorized aid on any assignment, quiz, or exam. A student cannot use the ideas of another and declare it as his or her own. Students are required to properly cite the original source of the ideas and information used in his or her work.

Application requirements

Implement a **Java-FX Matrix Calculator** application where you can perform the tasks described in the following. The application will basically let the user to transpose a matrix or perform three types of operations on two matrices of numbers.

Matrix Operations: Application Functions

- The overall goal of the application is to perform operations on matrices. Given two matrices (with numeric elements), A and B, the user should be able to compute the transpose A^T or B^T and: the Hadamard product A°B, the Matrix multiplication A*B, and addition A+B.

- These operations should be executed only if:

  - Transpose: it is always possible;

  - Hadamard product: the two operands matrices have the same number of rows and columns (https://en.wikipedia.org/wiki/Hadamard_product_(matrices));

  - Multiplication: the number of columns of A is equal to the number of rows of B (https://en.wikipedia.org/wiki/Matrix_multiplication);

  - Sum: the two matrices have the same number of rows and columns (https://en.wikipedia.org/wiki/Matrix_addition).

- The interface should let the user to:

  - R1: Enter the elements of the two input matrices in two areas (one side to the other) and visualize the result on a third output area, on the right of the input areas.

  - R2: Compute the transpose of a matrix entered in one of the two text areas and show the result on the output area. Hence, this operation should be possible, independently, for both input matrices.

  - R3: Select the operation to be performed between the two input matrices (Hadamard product, multiplication or addition);

  - R4: Select the number of decimal digits to be shown in the result;

  - R5: Compute the selected operation between two matrices when a button is pressed and display the result: a new matrix must be shown on the output area;

  - R6: Clear the content of the input text areas (by pressing the appropriate button);

  - R7: Display in a new window the history of all the operations. For each performed operation, this must include the operands (A and/or B), the operator (^{T}, °, * or +), and result (A^T, B^T, A°B, A*B or A+B) that have been computed since the application was started (current session only).

- In order to enter the elements of the matrices A and B you can use either two text areas (by separating the elements in the rows with a space and the rows with a new line), or two collection of text fields (one text field for each element of a matrix).

- In order to compute the transpose of an input matrix you can use a button.

- In order to implement the selection of the operation between two matrices and the choice of the number of decimal digits of the output you can use a combo box or radio buttons.

- In order to show the history of all the operations performed since the application was started (only the current session data) you must save the various input and output matrices (and the operation performed) in a string or in an array of strings and then show the content of the strings when the user requests that.

- The maximum matrix size could be limited to 10 x 10 (ten rows and ten columns).

- You can get **ideas** for possible GUIs for your application by looking at http://www.bluebit.gr/matrix-calculator/ https://www.calculator.net/matrix-calculator.html https://www.meta-calculator.com/matrix-calculator.php?panel-301-matrices But, pay attention to implement the requirements indicated in this document, these calculators are more complex than what is required here.

- The application must appropriately deal with possible bad input (and correctly notify the user in these cases):

    - R8: There are non-numeric data in the input matrices.

    - R9: The sizes (number of rows and number of columns) of the input matrices are not compatible with the requested operation.

    - R10: The numbers entered in a text area cannot be interpreted as a matrix (for instance the number of elements in two rows are not the same).

Grading

For each requirement that is not correctly implemented we will subtract a (fraction of) a point from the max grade (14 points).

We will also subtract one or more points (depending on the gravity of the issue) if:

- The GUI is not clear;
- The application stops working unexpectedly;
- The code is not well structured.