In this project, we will be creating some utility programs in java that meet the following specifications. In this assignment, you will have these specific source files. Nothing more, nothing less:

```
make_data_2d.java
read_data_2d.java
add_data_2d.java
```

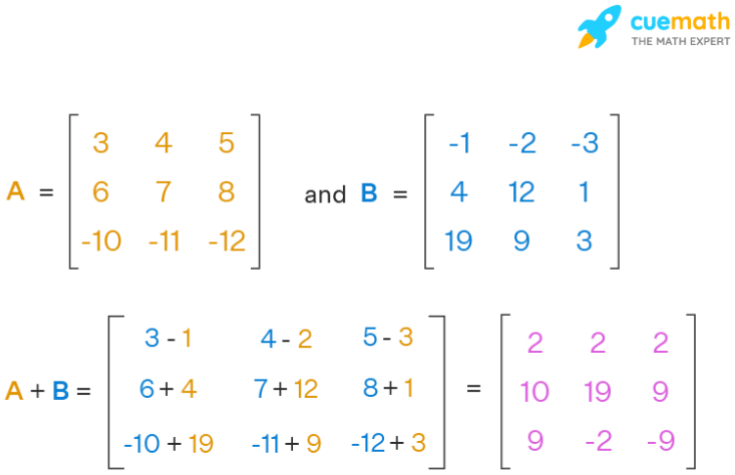
After making the project you will have two programs:

```
Usage: java make_data_2d <#rows> <#cols> <low> <high> <output
filename>
Usage: java read_data_2d <input filename>
Usage: java add_data_2d <in file1> <in file2> <out filename>

Where rows and cols are the number of rows and columns
associated with the 2D int (integer) array.  low and high
represent the lower and upper bounds of the uniformly
distributed data that will be randomly paced in each cell,
inclusive of the bounds, like the prior assignment.

(the data from the 2D integer arrays will be written and read
from the output and input data files, respectively, in BINARY
format, not string format) - So the files will not be human-
readable if opened in a text editor.
```

You'd be able to run it like this:

```
java make_data_2d 2 3 10 20 A.dat
java make_data_2d 2 3 5 15 B.dat

java read_data_2d A.dat

// here it would print a nicely formatted 2D matrix output to
the screen for the contents of A.dat, along with the time this
took in seconds (scientific notation, 1 decimal place. It should
also indicate the number of bytes per second that were written
to the file, same format.

java read_data_2d B.dat

// here it would print a nicely formatted 2D matrix output to
the screen for the contents of B.dat, along with the time this
took in seconds (scientific notation, 1 decimal place) It should
also indicate the number of bytes per second that were read from
the file, same format.
```

```
java add_data_2d A.dat B.dat C.dat

// This program would be the element-wise sum of the contents of
A.dat and B.dat and place the result in C.dat.  Note, this
program would not print C to the screen, but would print the
time this took in seconds (scientific notation, 1 decimal place)
It should also indicate the number of bytes per second that were
read and written from the input files and to the output file,
same format.  It should also report the number of additions
operations per second, in same format.


java read_data_2d C.dat

// Same as above description for read_data_2d.
```



Example of matrix addition (element-wise)

How to write and read binary data to / from a file.

Here is a ChatGPT link that shows some ways to accomplish this. I would recommend that you attempt to use the last example in each (DataOutputStream and DataInputStream). **NOTE: In the make_data_2d and add_data_2d, it is important that the these two program not be required to know or assume the size of the 2d arrays that are being read from the data files – like it shouldn't assume a certain number of rows or columns – they should work regardless, as long as the number of rows and cols are reasonable for the situation.**

NOTE – YOU MAY NOT USE ARRAY LISTS – YOU MUST ONLY USE 2D ARRAYS LIKE WE DISCUSSED IN CLASS

IMPORTANT: After completing the project – make sure you:

- Document your code well
- Test the programs thoroughly and test for possible mistakes and corner cases

Note, the add program must check for compatibility of addition (the dimensions must match).

REMEMBER, the files are to be written completely in binary – no strings / text!

Random number generation with "seed":

To make reproducible random number sequences, you need to seed the RNG. Here is a ChatGPT link with info about that.

Report:

Show your program working for *several different sized* matrices. Use the workflow above to show each step:

- Make a data file A.dat using make_data_2d
- Print A to screen using read_data_2d
- Make a data file B.dat using make_data_2d
- Print B to screen using read_data_2d
- Add A and B together to produce C using add_data_2d
- Print C to screen using read_data_2d

Show this in the report --- make sure it is COMPLETELEY clear that your code works.

Make a report called 2dReport.docx, and submit along with all documentation of what you did, the thought process, etc.

Again – make sure your code is documented – explain not only what everything is, and how it works, but why it works, and why it was done in the program.

Submission:

Complete all code
Document all steps of this, and the resulting operation of the programs. Demonstrate them working as described above.

Zip all files together (make clean first, also include one set of A.dat, B.dat and C.dat files that you used to test your program, the dimensions of these included data files should be 4 x 5. Make sure the report is in the folder structure and submit to Moodle.