CSC-361
Lab 3

Experimental Algorithmics with Matrix Operations

This lab asks you to implement, test, and time a scaled down `Matrix` class. Using empirical data acquired from your executions, you will analyze two operations on the Matrix class to determine a linear regression model.

Matrix Class

The `Matrix` class will store doubles and implement the operations described in Table 1.

Table 1: Methods to Implement for the `Matrix` class

Return Type	Method	Description
`Constructor`	`Matrix(int rows, int columns)`	Constructs a `rows x columns` matrix containing only `0`s.
`Constructor`	`Matrix(List<Double> items, int rows, int columns)`	Constructs a `rows x columns` matrix containing the primitive `double` values from `items`. If more values are given in the list than needed, they are ignored. If fewer values are given than required, the rest of the matrix will contain `0`s. This method is useful for unit testing purposes.
`static Matrix`	`create(int rows, int columns)`	Constructs a `rows x columns` matrix of values between `0` and `1` (using `Math.random`).
`Matrix`	`plus(Matrix that)`	Constructs and returns a new `Matrix` object representing `this + that`. When the dimensions of `this` and `that` are not equal, a `RuntimeException` is thrown.
`Matrix`	`times(Matrix that)`	Constructs and returns a new `Matrix` object representing `this x that`. For a valid matrix multiplication operation, a $\texttt{rows}_{\texttt{this}}$ x $\texttt{columns}_{\texttt{this}}$ multiplied by $\texttt{rows}_{\texttt{that}}$ x $\texttt{columns}_{\texttt{that}}$ results in a matrix that is $\texttt{rows}_{\texttt{this}}$ x $\texttt{columns}_{\texttt{that}}$. If $\texttt{columns}_{\texttt{this}}$ `!=` $\texttt{rows}_{\texttt{that}}$, then a `RuntimeException` is thrown.

Timer Class

To facilitate timing a `Timer` class has been provided. It should be used in `main` something like the code below.

```
Timer timer = new Timer();
timer.start();

left.times(right); // The method we are timing (left and right are Matrix objects)

timer.stop();
System.out.println(rows + " " + columns + ": " + timer.toString());
```

For data collection purposes, you might consider capturing the time (in milliseconds) that `stop()` returns and using that in your analyses below.

Generating Data

Our goal is to see if we can experimentally approximate the theoretical efficiency of matrix addition and matrix multiplication; to do so we must collect some runtime data for each algorithm.

In a `main` method in a `Main` class, implement a looping strategy to accumulate runtimes of increasing sizes of square matrices *for each* method. When acquiring final data, execute your code with as few programs running in the background as possible. This will result in cleaner data since it is less likely for a thread to interrupt your code resulting in outlier values.

Feel free to output your timing data to the console (as shown above).

JUnit Testing

You are to implement a small set of unit tests to verify matrix addition and matrix multiplication. ***No output*** should be produced by your tests; we are seeking only a ‘green’ output indication in Eclipse.

Submitting Source Code

Your code should be well documented, including docstring comments of methods, blocks of code, and header comments in ***each*** file.

Testing code needs fewer comments as they should be self-descriptive; however, it is recommended that each individual test or family of tests be numbered and have a brief comment.

Header Comments

Your program must use the following standard comment at the top of *each source code file*. Copy and paste this comment and modify it accordingly.

```
/**
 * Write a succinct, meaningful description of the class here. You should avoid wordiness
 * and redundancy. If necessary, additional paragraphs should be preceded by <p>,
 * the html tag for a new paragraph.
 *
 * <p>Bugs: (a list of bugs and / or other problems)
 *
 * @author <your name>
 * @date   <date of completion>
 */
```

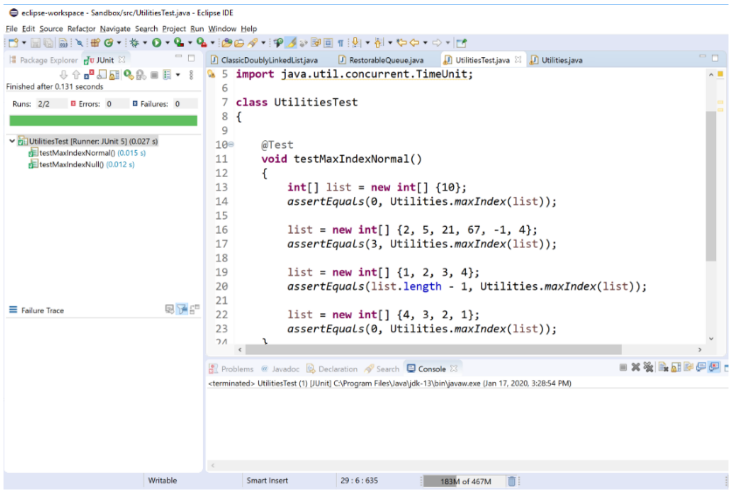
Inline Comments

Comment your code with a *reasonable amount of comments* throughout the program. Each method should have a comment that includes information about input, output, overall operation of the function, as well as any limitations that might raise exceptions; Javadoc comments are ideal. Each *block* of code (3-4 or more lines in sequence) in a function should be commented.

It is ***prohibited*** to use *long* comments to the right of lines of source code; attempt 80 to 100 character-wide text in source code files.

Submitting; Proof of Program Execution

Execute your code and take a screenshot of the associated output console. Place these screenshots into a word processing document (Word, OpenOffice, GoogleDocs, etc.). If multiple screenshots are necessary, label each clearly. Please make sure to crop and enlarge the screenshots so that the picture and / or text is clear (and doesn’t strain my old eyes). For example, ***the screenshot on the next page is not appropriately sized*** although it contains ideal information (output console, code, etc.). Create a PDF of this document and call it `evidence.pdf`.



Finished after 0.131 seconds

Runs: 2/2 Errors: 0 Failures: 0

```
UtilitiesTest [Runner: JUnit 5] (0.027 s)
    testMaxIndexNormal() (0.015 s)
    testMaxIndexNull() (0.012 s)
```

```java
import java.util.concurrent.TimeUnit;

class UtilitiesTest
{

    @Test
    void testMaxIndexNormal()
    {
        int[] list = new int[] {10};
        assertEquals(0, Utilities.maxIndex(list));

        list = new int[] {2, 5, 21, 67, -1, 4};
        assertEquals(3, Utilities.maxIndex(list));

        list = new int[] {1, 2, 3, 4};
        assertEquals(list.length - 1, Utilities.maxIndex(list));

        list = new int[] {4, 3, 2, 1};
        assertEquals(0, Utilities.maxIndex(list));
```

```
<terminated> UtilitiesTest (1) [JUnit] C:\Program Files\Java\jdk-13\bin\javaw.exe (Jan 17, 2020, 3:28:54 PM)
```

Source Code Files

You are to submit your entire project folder (including any files provided to you).

Final Submission File

In a folder named `lab`, place (1) the project code folder and (2) `evidence.pdf`. Zip folder `lab` and label that zip file as `lab.zip`. This zip file is to be submitted via Moodle.