EECS 2030: Lab 3

(about 3 % of the final grade, may be done in groups of up to three students)

Due: as set in *eClass*

Motivation

This lab will ask you to do the following:

- use arrays as fields for two immutable classes.
- complete the constructors of those two classes
- complete the implementations of the overridden versions of the `equals`, `hashCode`, and `toString` methods in one of the classes
- complete several accessor methods.
- implement a simple utility class using the two classes above

Getting Started

Download a zip file containing the Lab 3 Eclipse project.

Import the project into eclipse by doing the following:

1. Under the **File** menu choose **Import...**
2. Under **General** choose **Existing Projects into Workspace** and press **Next**
3. Click the **Select archive file** radio button, and click the **Browse...** button.
4. In the file browser that appears, navigate to your download directory (exactly where this is depends on what computer you working on; on the lab computers the file will probably appear in your home directory)
5. Select the file **lab3.zip** and click **OK**
6. Click **Finish**.

Part 1: Immutable Classes

Implement classes `Vector3` and `Matrix3` in the package `eecs2030.lab3` that represent immutable classes for a 3-component vector and a 3 × 3 matrix. See the documentation contained in the class files.

- Try to use constructor chaining to implement the required constructors whenever possible.
- Think about how the immutability is ensured whenever you implement any methods or constructors
- If you cannot complete one or more of the methods, at least make sure that it returns some value of the correct type; this will allow the tester to run, and it will make it much easier to evaluate your code. For example, if you are having difficulty with `hashCode` then make sure that the method returns some numeric value.
- You are encouraged to explore the features of the `Arrays` class, in particular `copyOf`, as well as the `System` class – `arraycopy`.

JUnit testers for your classes are available in the project that you downloaded. Note that the testers are not very thorough, and may not catch all the errors you might make. Also note that passing all of the tests in this tester *does not* guarantee a good solution (in other words, you should think critically about your implementation for each method).

Part 2: Utility Class

Complete the implementation of the `MVMath` class. Refer to the method's description in the JavaDoc in-code comments. In case you need to refresh your knowledge of vectors, matrices, and operations involving them, feel free to consult your favourite math textbook or the numerous online resources.

A JUnit tester class for this part contains only two methods (testing matrix-matrix multiplication). Complete the other unit tests to be able to test the other newly implemented methods.

If you have questions, don't hesitate to post your questions on the course forum on eClass.

Submission

Find all the `java` files in your project (there should be six of them) and submit them electronically via eClass (do not zip them).

If working in a group, make only one submission and include a **group.txt** file containing the names and the student numbers of the group members. The deadline is firm.

Grading

The assignment will be graded using *the Common Grading Scheme for Undergraduate Faculties*[1]. We look at whether the code passes the unit tests, satisfies the requirements of this documents, and whether it conforms to the code style rules.

Academic Honesty

Direct collaboration (e.g., sharing your work results across groups) is not allowed (plagiarism detection software may be employed). However, you're allowed to discuss the assignment requirements, approaches you take, etc. You may not use any code from outside sources, even if it is your own code you wrote for another assignment, project, hobby, etc.

[1] https://secretariat-policies.info.yorku.ca/policies/common-grading-scheme-for-undergraduate-faculties/