CSC 115: Fundamentals of Programming II
Assignment #1: Multiple Classes, Tester

Due date

Monday, May 29th, 2017 at 9:00 am via submission to conneX.

How to hand in your work

Submit the requested files through the Assignment #1 link on the CSC 115 conneX site. Please make sure you follow all the required steps for submission – *including confirming your submission!*

Learning outcomes

When you have completed this assignment, you should have become (re-)acquainted with:

- how to create a *one-dimensional (i.e., 1D) array*;
- how to *read from and write to array elements*, using both *explicit and computed index values*;
- how to *create instances* of custom classes;
- how to *compile*, *execute* and *test* a multi-class program;
- how to use *visibility modifiers* such as *private* and *protected*.

In addition to these you will be introduced to:

- how to *read and use a method specification*.

The task for this assignment

Problem description

For this assignment you must complete the implementation of some methods contained within three provided classes in a way that satisfies the specifications given for methods in the classes. The three classes are:

- *Student.java*
- *GradStudent.java*
- *Course.java*

A fourth class has also been provided:

- *A1test.java*

and its purpose is to use the results of your work. That is, *A1test* will test your work in *Student*, *GradStudent* and *Course* for basic correctness.

Therefore one of the first steps you should take after downloading all four source-code files is to compile and run the test program. Assuming all files are in the same directory, you will do this:

```
$ javac A1test.java
$ java A1test.java
Testing of 'Student' class (basic)
Failed test: 1 at line 73
```

The tester program indicates here that your implementation – which, of course, has yet to be started! – fails in the very first test. That test was exercised at line 73 in *A1test.java*. If you read this line and the few before it, you'll see that line 73 is a check on whether or not the student ID given in the constructor for a student object is indeed what was returned by a call to *getId()* on that object.

The purpose of some tests will be quite clear and obvious, some will require a bit of thought to understand, and others are more complex only because of the code used to implement the test (i.e., the "stress" tests). However, reading code is as important a task as writing code. Therefore by reading the tests as they pass (and fail) along with comments provided before the empty methods in *Student*, *GradStudent* and *Course*, you'll slowly but surely determine what is required to pass all tests.

A word of caution: It is possible for some tests to pass at first only to then fail after additional code later completed. This is not unusual and is often a sign that your existing code has a bug that is only now exercised by what the newer code has constructed. Once you find and fix the bug, the earlier test will again start to pass.

When we evaluate your work, it is the first failed test that will matter most for us (i.e., we do not cherry-pick test by ignoring earlier failed tests while accepting later passing tests). Tests after the first failed test are themselves considered to have failed.

There are ~~51~~ 52 tests and your code is expected to pass them all.

Suggested implementation strategy

1. Start your work by attempting to complete *Student.java*.
2. First implement the constructor that takes two parameters.
3. Then implement *getCredits()*, *getName()* and *getId()*. Recompile *Student.java*. (From this point onwards the text will not indicate when you should recompile. I assume you can determine this for yourself.)
4. Now re-run the tester. Assuming your work is correct, you will have passed tests 1, 2 and 3. The first failure will now be test 4 (i.e., on line 78).
5. Implement *setCredits()*. Again assuming your work is correct, you will have passed the first four tests. The next failure, however is an exception – you will be told a *NullPointerException* occurred at line 81 in *A1test.java*.
6. If you look at that line, you'll see it occurs right after a call to the *Student* constructor that takes three parameters. Given what you've written so far, why might this be the case?
7. Assuming you have determined what caused the error in step 6 and written the appropriate additional code, the first seven tests will pass, and now the first failing test is test 8.
8. Implement *setName()*. Re-run the tester, and now the first failing test will be 10.
9. Implement *equals()*. (Be sure the read the specification carefully, and remember comparing strings with "==" can often give false negatives.) After this is correctly completely, the first failing test will be 12.
10. Implement *toString()*. After this is correctly implemented, all of the first set of tests will have passed (i.e., the first 15 tests).

At this point you will have completed the implementation of *Student.java*. Proceed with *GradStudent.java*, and then *Course.java*. (Hint: When implementing a class always first complete its constructors.)

Note that some tests are more complicated than others.

Files to submit (three in total):

a. *Student.java*
b. *GradStudent.java*
c. *Course.java*

Submit these using conneX. Ensure you actually submit the files and do not just submit a draft assignment.

Grading scheme

Requirement	**Marks**
All three java files compile without errors or warnings	2
Code passes the first set of test cases (1 to 15)	2
Code passes the second set of test cases (16 to 28)	3
Code passes the third set of test cases (29 to 33)	3
Code passes the fourth set of test case (34 to 42)	3
Code passes the fifth set of test cases (43 to 49)	2
Code passes the sixth set of test cases (50 to 52)	3
Code uses follows the posted coding conventions on commenting.	2
Total	**20**

Note: Code submitted must be written using techniques in keeping with the goals of the assignment. Therefore passing a test is not automatically a guarantee of getting marks for a test (i.e., your solution must not be written such that it hardcodes results for specific tests in *A1test.java* yet would be unable to work with similar tests when different data is used).

In order to obtain a passing grade for the assignment, you must satisfy at least the first four requirements by passing all of their test cases.