Advanced Programming 1 - 89-210 - Ex2

November 26, 2016

Due date: 4.12.16

Intro

In this exercise, we will use the initial design we made last exercise, and prepare unit tests as we learned in class. This approach of writing down tests first and only then the code itself is called TDD. Thinking about the way you'll test your code before you even write it, helps preventing bugs and issues you would get only after writing down the code.

Implementation

Review the UML diagram you designed last exercise. For every class generate two files: .h and .cpp. Inside the .h files write down the class signatures. Inside the .cpp file add the corresponding methods, without any code implementation inside. For methods which have a return type which differs from *void*, return some default value (e.g. 0 for int, 0.0 for double, NULL for any non primitive object etc...)

Now what's left to be done, is to implement unit tests for every method.

- Notice that you can't test private and protected methods as those aren't accessible from within the tests scope. This is ok, as those methods are meant to be used internally. This doesn't mean you don't need to test those methods logic, it only means you can't do that explicitly. So make sure to test the public method which use these internal methods. You should even dedicate extra attention to the public methods which make use of protected and private ones, as those are (at least) slightly more complicated.
- Every public method must contain at least one test method / function.

Recommendation: Think before writing! Try to figure out the different scenarios which can take place. After imagening all possible scenarios - general cases and edge cases - start writing down the tests.

Submission:

All source code you wrote with a makefile which compiles your code, zipped. We will not run the code (as it's not implemented yet), but in the next exercise, it should run perfectly and pass all the tests you wrote. But we will compile it (actually the submit system will).

You will be tested on your test coverge - that is, on what percent of the public method you wrote tests, and the quality of those tests - meaning how deeply you thought on different cases the code needs to handle.

Notes

- Don't be late.
- Submit your work to the submit system, with your personal user account (once per couple with the file which specifies the members, as described in the submission document).
- As always, submit a details file along of your files (specified in the submission notes file).
- For using google test plugin in eclipse: eclipse-plugin, or in Clion: clion-plugin.
- The google library is installed on the u2 and on the submit system, so it should work correctly. Make sure before you submit that it does indeed compile as expected on the u2 server.
- Again, we will not run your tests, as it won't pass. But it does need to pass the compilation phase.