CSCE 740 - Project 5 - Testing

Due Date: Monday, December 7, 11:59 PM (via Moodle)

Overview

We've built GRADS. Time to kick back with one of those little drinks with an umbrella in it.

Wait! We should probably test it a little before it ships.

The Assignment

You will create automated system-level tests for your GRADS implementation and measure the code coverage achieved by your tests. By system-level, we mean tests will test GRADS as a **black box** – by using the functions offered by the top-level GRADS object – not the other classes defined in your design document. Thus, all tests will provide input to GRADS through the interface we provided, and will verify that the output matches your expected output.

For example, if you want to write tests related to generating progress summaries, your tests need to:

1. Create a sample student record where each field has a value, add its information to the JSON file.
2. Create a ProgressSummary instance filled with the expected values for each of its fields.
3. Create a GRADS instance.
4. Load the student record JSON file.
5. Log in as a valid user (calling `grads.setUser(userId)`).
6. Generate a progress summary for the student defined in the student record (calling `grads.generateProgressSummary(studentId)`)
7. Check whether the generated summary matches the expected outputs defined in step 2.

You will need to have completed homework 4 to complete this assignment. If it is incomplete, finish it now.

- **Task 1:** Update and implement your requirements-based tests as JUnit test cases. Remember, you will need specific, concrete inputs – for example, when generating a progress summary, we expect you to explicitly state all information necessary for the method call including every data value that should be present in the student record – as well as concrete expected outputs for each test. The test case must be complete, covering all details of the test.

  Test case implementations should use descriptive names, both for the classes and the methods you implement. We do not want to see test1(), runTest4(), etc. Implementation

of test cases must follow provided coding standards (see the assignment 4 description for details).
- **Task 2:** **Measure the statement/line coverage achieved by these tests (include this report with the ZIP you hand in).**
- **Task 3:** **Supplement your requirements-based tests with additional unit-level tests until you achieve at least 80% statement/line coverage.**

Tools

Completing this project requires writing tests in the JUnit testing framework and measuring statement/line coverage with the **EMMA coverage tool (also available as an Eclipse plugin called eclEmma)**.

We will not provide tech support for tools. Read the documentation and ask questions of your peers on the Moodle forum. Links to tools are available on the Moodle course page.

Deliverables

You are responsible for delivering the following as a single zip via Moodle:
- Requirements-based system-level tests, written as executable JUnit code.
    - If updated, also submit your revised project 4 code.
    - Tests should not be mixed with project code, but should be in their own package.
- Coverage report from executing these tests.
    - EMMA provides HTML reports.
- Additional unit-level tests to achieve 80% statement/line coverage.
- README document describing any additional information needed to run tests (i.e., if you used any additional tools or libraries, such as mocking frameworks)