CSE 321 – Homework Assignment #2

Preamble

You will complete this homework in small groups, assigned randomly by Canvas. Please check Canvas for your group.

Turn in a PDF and other files to canvas by the deadline. It can be a scan of handwritten work, if that is your preference. But it must be legible in that case!

You may post questions to the related Discussion board, but do not share any of your work/solutions.

Please review the course syllabus and first lecture for a reminder of acceptable collaboration (outside of your group) and unacceptable collaboration. Ask me if you're unclear.

Start early!

Contribution Declaration

You must describe each person's contribution as a separate section of the homework. Any evidence of unfairness will result in a grade penalty.

Office Hour Reminder

- Liam Mosley, mosleylm@miamioh.edu , Office Hours Wednesday 7-830pm in Benton 02
- Emily Richmond, richmoet@miamioh.edu , Office Hours Thursday 5-630pm in Benton 08
- Dr. Stephan, matthew.stephan@miamioh.edu, Tuesday 1:15pm – 2:35pm; Thursday 2pm – 315pm

CSE 321 Homework #2 Due February 28th, 11:59pm

Questions

Question 1 (18 points)

Find Calc.java on the Canvas page, indented from this homework.

Calc currently implements one function: it adds two integers. Use test-driven design in JUNIT (tests first!) to add additional functionality to

1. subtract two integers,
2. multiply two integers,
3. and divide two integers.

First, create a failing Junit test for one of the new functionalities. Then, modify the class until the test passes. Lastly, perform any refactoring needed. Repeat until you have added all of the required functionality to your new version of Calc, and all tests pass.

Remember that in TDD, the tests determine the requirements. This means you must encode decisions such as whether the division method returns an integer or a floating point number in automated tests before modifying the software.

Submit printouts or submissions (.java files) of all tests, your final version of Calc, and a screen shot showing that all tests pass. Most importantly, include a narrative describing each TDD test created, the changes needed to make it pass, and any refactoring that was necessary.

(2 points x 3 sets of Tests for each functionality piece. Screenshot of "failed tests" included).
(2 points x 3 correct implementations of the functionality of Calc)
(1 point screen shot of tests passing)

(5 points for the narrative: 2 points – Description of TDD tests. 2 points – Changes needed to make it pass. 1 points – any refactoring that was necessary/you did)

Question 2 (8 points)

Set up a continuous integration server (localhost/locally is fine or you can use online solutions). Include version control for both source code and tests, and populate both with a simple example. Experiment with "breaking the build", by either introducing a fault into the source code or adding a failing test case. Restore the build.

1 point – Demonstrate (via screenshot or link) setup of CI server
1 point – Demonstrate (via screenshot or link) version control solution
2 points – Example with both source codes and tests
2 points – Demonstrate how you broke the build and what the fault/failed test case is. That is, show an example of you breaking the build and a fault/failed test case.
2 points – Proof that you restored the build (before picture/image, the command you used, and after picture/image).

We will not be assisting you in setting up these systems as 1) this is the learning outcome for this question 2) we may not be familiar with the CI systems you are using, so are unable to help.

Example CI Servers Systems

- https://jenkins.io/ is popular and good for your resumes. Can be difficult to setup
- https://travis-ci.org (recommended by Emily: "It's web based, auto syncs with GitHub, and lots of getting started docs.")
- https://about.gitlab.com/features/gitlab-ci-cd/
- And more - https://stackify.com/top-continuous-integration-tools/

Question 3 (8 points)

Suppose that coverage criterion C1 subsumes coverage criterion C2. Further suppose that test set T1 satisfies C1 on program P and test set T2 satisfies C2, also on P .

(a) Does T1 necessarily satisfy C2? Explain. (1 point answer, 2 point explanation)

(b) Does T2 necessarily satisfy C1? Explain. (1 point answer, 2 point explanation)

(c) If P contains a fault, and T2 reveals the fault, T1 does not necessarily also reveal the fault. Explain. (2 points)