**COSC 603: Software Testing**

**Assignment 3**

**Due Tuesday, 02/25 (by the start of class)**

**Objective:**

This assignment helps you practice TDD, CI and regression testing.

**Tasks:**

**Configuration of GitHub**

1. If you do not have a GitHub account, please sign up a free GitHub account. It is recommended to use your edu email, because GitHub is providing some useful tools through GitHub Education package (please check details at: <https://education.github.com/pack>). If you already have a GitHub account, please go to the next step.
2. Please go to my repository at <https://github.com/dmark1021/Travis412>. There you will see some source code and configuration files. Then, fork the repository to your own account by clicking the “fork” button on the top right corner. After several seconds, you should see everything in your own repository (it would be better to take a screenshot here to show this milestone).
3. Click the green “Clone or download” button on the middle right of the screen and copy the link shown in the small panel. For example, if my account is **cosc603**, the link should be: <https://github.com/cosc603/Travis412.git>. (this is just an example, you will have your own unique link for it.)
4. Come back to your local computer, launch a terminal (Mac) or command line prompt window (Windows). Go to a folder you would like to store all your code for this assignment by using the command **cd**. Type the following command to clone the repository from GitHub:
5. git clone <https://github.com/cosc603/Travis412.git> (this is just an example, you will have your own unique link for it.)
6. After that, you should be able to see source code downloaded from GitHub to your folder.

**Import to Eclipse (option 1)**

1. Launch your Eclipse. One the left side you will see “Package Explorer” pane. Right click on the empty space in this pane, click “import…”.
2. Choose “Maven -> Existing Maven Projects”, then “next”.
3. Click “Browse…” on the top and choose the source code folder just downloaded in step 5. Make sure pom.xml file is checked. Click “Finish”. Then you should see the project is successfully imported into Eclipse. You should also see that the GitHub information is tracked by Eclipse with showing “[Travis412 master]”. (it would be better to take a screenshot here)
4. In this project, there is a template class *SimpleCalculator.java* and a test file *SimpleCalculatorTest.java*. Right click the project, then select “Run As -> Junit Test”, you should see that the test is executed and passes with a green bar.

**Import to IntelliJ (option 2)**

7. If you would like to use IntelliJ, in the menu, click File -> New -> Project from Version Control -> Git.

8. Log in to GitHub using your GitHub ID.

9. Choose the repo Travis412 you just cloned, then click Clone.

10. You may need to mark test folder to be “Test” in Open Module Settings. Please make sure you can run the test file *SimpleCalculatorTest.java* and see the testing results.

**Configuration of Travis-CI**

1. Launch a browser and go to <https://travis-ci.org>. (Make sure it is org not com, because org is free version.)
2. Click the button “Sign in with GitHub” on the top right corner. The first time you sign in, it will ask you to authorize. After authorization, you should see your GitHub repository here in Travis-CI.
3. Enable the Travis-CI support by moving the option bar after the repository name from left to right and it should show as a check.
4. Go back to your GitHub repository page, and locate the file *.travis.yml*. This is the file that specify configuration information for Travis-CI. Click on the file and you will see the content of it. Click the small pen icon for “Edit this file” on the right side of *Raw, Blame, and History*. Change the email address to your own email. Do not change other settings. After that, click the green “Commit Changes” button at the bottom to commit this change.
5. Once you change your file on the GitHub website, you need to go back to your Eclipse or IntelliJ to sync the changes back to local.

**Try Pushing Something**

1. To test whether they all work together correctly, in your Eclipse, change the test file SimpleCalculatorTest.java by modifying the result of 1 + 1 from 2 to 0. Run this test you should see a red bar in Eclipse.
2. Right click on the project, click “Team -> Commit”. You should see a window for GitHub in Eclipse. In the window of “unstaged changes”, there are files that you changed but not synced to the GitHub repo. Click those .java files then click the + sign to stage them. Then, write some commit message to show why you want to change it. After that, click “Commit and Push…” button. Your changes will be pushed to the GitHub repository.
3. Ideally, Travis-CI should automatically detect this change and re-build the entire project then re-test it using the tests in the repo. Of course, we just upload a failing test in the repo, so the building and testing process will fail. The email you set in step 13 should receive an email notification telling you that something is wrong in your repo in several minutes.
4. Fix the failing test in your Eclipse then push it to GitHub again following step 15-17. You should receive another email saying everything is fixed. Also, the small icon on the GitHub repository page indicates the real time status of the repo by showing whether it is passing (green) or failing (red).

**TDD Process**

1. Then, the last set of tasks are to add the features of minus, multiply, and divide to the existing *SimpleCalculator* class following the TDD process. Each feature should be a separate method in this class similar as the one for add.
2. First, without changing anything in the *SimpleCalculator* class, add one more test that performs the minus function in *SimpleCalculatorTest*. For example, compute 5 – 3 and expect a 2. You can follow the existing test example. It will be surely a compiling error but don’t worry.
3. Push it to your GitHub repo, and your Travis-CI should complain by sending you an email saying you break something.
4. Come back to the *SimpleCalculator* class, add the feature of minus, and make the test you add in step 20 pass locally. Then, push it to GitHub. Your Travis-CI should notify you that you fixed it.
5. Adding the features of multiply and divide by following step 20 – 22. Make sure taking screenshots during the process to indicate you follow the TDD process.
6. Practice and add more tests, for example, considering the divide by zero problem.

**Submission: Please submit your assignment on Blackboard.**

Grading Rubric: [Total: 20 points]

Submit a brief report in two parts:

(10 points) — Source code

(7 points) source code and JUnit tests.

(3 points) Evidence that your JUnit tests run (such as screenshots)

(10 points) — Write-up

(5 points) Screenshots and discussion about how you use these tools to finish the continuous integration.

(5 points) Discussion of using TDD process. For example, how you think it is different from the traditional software development process. Any drawbacks or advantages, etc.