Assignment 4 - Code Review

Version: February 13, 2018

Due: Monday Feb 19th, 11:59pm

Objectives

- Understand how to review code and to integrate reviews into source code control.

General

This task is meant to be a team task for 2 students. See below where it states exactly what you are supposed to do.

I included a new Gradle file in here that runs JUnit 5. The test classes are also now JUnit 5 test classes. This is to show you how you could use JUnit 5, since it was not used in the other assignment. I also included tasks in Gradle which create HTML reports again. In your project you can use JUnit 4 or 5.

Task 1: Perform Code Review (10 points)

1. (individual) For this assignment grab the zipfile from Blackboard. Expand the zipfile in an empty folder on your computer.
2. (one of you) Create a GitHub repo with your copy of the code. Name the repo asurite_review
3. (one of you) add user "ser316asu" to your project
4. (one of you) add the student you are working with to the project
5. push the review file to GitHub as Word document (the one given on Blackboard)
6. (individual) Import the project into Eclipse.
7. (individual) Review the source code files for conformance to the defects provided in the coding standards document (Coding standards, for general quality practices - Code Smells, and for logic errors). Log any defects using the MS-Word form. You (individually) should at least log 10 defects in the given log file. Be sure to provide the information in that form (ID, Location, Problem description Category (CS, CG, FD or MD - defects that do not fit in the other categories). You should log at least one defect for each of the first three categories (CS, CG, FD). It does not count as different defects if you state CS 1 a couple of times (file header). Make sure you find different errors.
8. (individual) when you are done commit and push your review form to GitHub (overwriting the old one)
9. (individual) when the second one is done you should merge and also push to GitHub! You can resolve conflicts together.

Task 2: Apply the Git Pull Workflow (15 points)

1. (team) each of you select 3 defects from your Word document from Task 1 (they have to be from different categories).
2. (individual) Create a branch for each of these defects. Name them bug-# where #s are the number identifiers of the defects on your code review form.
3. (individual) Pick one bug and resolve the defect locally on your computer, in the correct branch.
   a) Push the change to the proper remote branch on GitHub.
   b) Create a Pull Request to merge the branch into the master branch.
4. (individual) Repeat the steps in 3 for the second and third bug.
5. (team) Your team member needs to review and accept or decline the PullRequest. Each of you thus has to check 3 PullRequests. You should decline and accept at least one request.

DO NOT DELETE YOUR GITHUB BRANCHES AFTER CLOSING THE PULL REQUESTS!!! WE NEED THESE TO GRADE YOUR LAB!!!

Submission

Submit the link to your GitHub repo with the name of your partner (only one of you submits).

Going Forward:

I expect that from this point forward you will incorporate code reviews into your quality policy:

1. Decide on when and where and why you will do reviews. Which code? How many people? What level of formality? I expect each team to define one formal and one informal code review process in their quality policy and post this on their Taiga Wiki (due when this assignment is due).
2. Define how you will categorize issues – level of severity, type of defect, etc. How will you indicate that?
3. Define a coding standard. Quality codebases look like one person writes them.
4. Integrate your process into GitHub. I expect to be able to track code reviews you have performed via GitHub. I should clearly see pull requests with labels/comments based on code reviews. Follow the Git Pull Workflow.
5. Make sure in your process it is clear whose code is getting reviewed, and by whom it is getting reviewed. You should serve as both an author and a reviewer to get credit for this module on your project.

In your project

From now on you should update your Quality Policies to include Reviews. No code should be checked into Master without a review. Even documentation changes should be reviewed. Code going into Master should be looked at by at least two team members.

In your project you can choose between JUnit 4 and JUnit 5. In my opinion JUnit 4 is a bit easier to handle and generates the HTML reports without a lot of overhead. But this is absolutely up to you. Just make sure you decide **in your team** what you want to use! Let me know in your Quality Policy what you use.

You should make sure that all your tests are in one folder (preferably src.test.java since this is where Gradle expects them to be).