

Teacher's Name: Munawar Hafiz

This is an individual assignment. Everybody has to submit

We want to hit two targets (with one stone/assignment).

1. Identify code smell in each others code and fix them (Good Coding Practice).
2. Experience code review using Gerrit and git.

Step 1: Good Coding Practices:

This is copied from the previous assignment specification. All the instructions remain the same. The last time, I wrote code and you picked issues in my code. This time you will be reviewing each others code, putting review comments in their code, and make sure you fix them.

You have seen the video discussing the good coding practices from the Clean Code book.

<https://www.youtube.com/watch?v=HZJxjlvBbVA>

You also have access to the good coding guidelines.

(Posted under 16-Coding Guidelines, Document names starting with 00 and 01... One is general good coding guidelines from the Code Complete book, the other is coding guidelines for Java programmers)

Follow these guidelines.

Step 2: Code Review

We will do collaboration at a massive level. Every student will sign up as a reviewer in 7 other projects. On the other hand, every student's code will be reviewed by 7 other students.

Each of you will put the code you wrote in CA05 up for review. That code was produced as a pair programming project. So, two people will use the same project in this assignment. As a student, your responsibility will be to avoid signing up (as a reviewer) to the same project twice. How will you know? We will create a list of the pairs on a Discussion Entry on Canvas. When you sign up to review 7 projects, make sure you are not selecting two of the same projects done by a pair. Do not try to bypass this. We will check and you will get a **zero** if you do it. At the same time, you will be very very bored if you did this, since you will be forced to repeat your work twice.

Here are the steps you will follow. I will use the terms **project owner** and **reviewer** in the rest of the document. You will have to assume both roles in this project. As usual, there is the Excel file that you will have to fill up describing your activity.

Goal:

- You will be signing up with 7 other projects as a reviewer. You will be making at least 7 code review comments for each project you sign up. A total of 49 comments have to be made by you at least. Since you will be racing with others to find code smell, being an early bird will allow you to catch more worms.
- You will be receiving at least 49 code review suggestions as the project owner (7 each from the 7 people assigned to your project). You will have to go through the review sequence, until you are allowed to incorporate the code in your repository.

Tools:

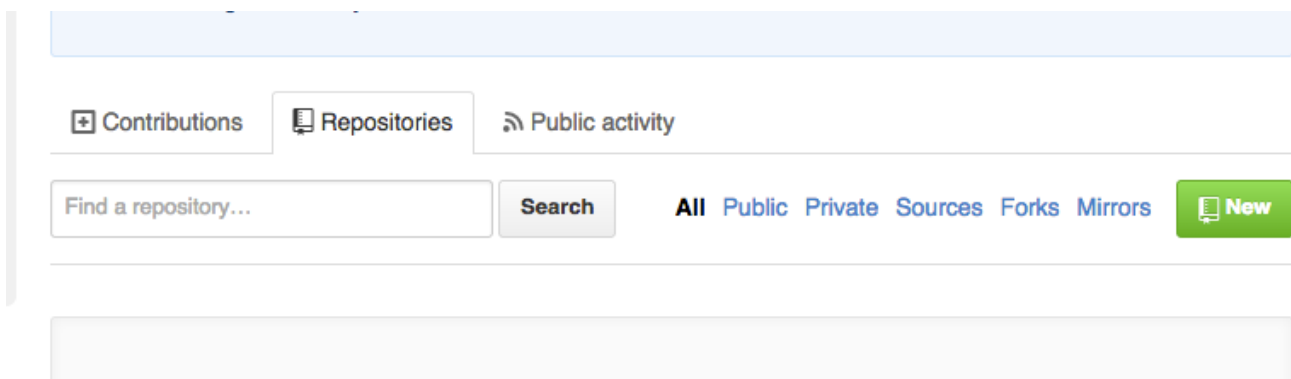
We will be using GitHub and GerritHub, two web based tools. No other tools are necessary. You just need to create an account on GitHub. GerritHub uses your GitHub account.

Coordination:

GitHub and GerritHub has some limitations for what we are doing. They are meant to be useful if you are developing code and reviewing code at the same time. Here, we have the code already developed. We will only do code review. That is why we need to start together, and start early. **If you start late, you will be holding potentially the entire class back with you (because you will be collaborating with pretty much the entire class as a reviewer or a project owner). We will monitor and deduct points if you make delay. Your friends will also not let you go with it.**


Here are the steps:

- Project owner will create an empty repository on GitHub.



- b) Make sure that the project is public. Also, make sure you check the initialize this repo with a README checkbox. The second one is not mandatory, it will just make the file creation obvious.

Owner

 munaha2 ▾


 /

Repository name


stockmarket

Great repository names are short and memorable. Need inspiration? How about...

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you don't want to.

Add .gitignore: **None** ▾

 |

Add a license: **None** ▾ 

Create repository

- c) Project owner will manually enter each of the project files in proper directories in GitHub. You cannot do automated porting, even though people will typically do it that way. This stipulation is because you have to add files and create “Pull Requests” that GerritHub will use to track your project (Explained later).
- d) To create a file, hit the plus sign beside the project.

The screenshot shows the GitHub repository creation interface. At the top, there is a text input field for a 'Short description of this repository' and a 'Webs' button. Below this, a summary bar shows '1 commit' and '1 branch'. A green 'Commit' button is visible, along with a dropdown menu for 'branch: master' and a link to 'stockmarket / +'. The 'Initial commit' section shows a commit by 'munaha2' just now, with a file named 'README.md' listed as the 'Initial commit'.

- e) Add appropriate directory structure following your project, copy paste the content in the box.
- f) At the bottom of the page where you will push commit, change the radio button to the second one which asks for creating a pull request. On the following page, just click on create pull request again. **This is the most important step.** It is critical to do this to ensure GerritHub will coordinate with each other. **Enter all the files in this manner. There will be several pull requests.**

The screenshot shows the 'Commit options' dialog in GitHub. It has a text area for 'Add an optional extended description...'. Below this, there are two radio buttons: 'Commit directly to the master branch' (which is unselected) and 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' (which is selected). A text input field below the radio buttons contains the branch name 'munaha2-patch-1'. At the bottom, there are two buttons: 'Propose new file' (green) and 'Cancel' (grey).

In case you are wondering what is a pull request, read here:

<https://help.github.com/articles/using-pull-requests/>

Pull requests are notification to collaborators that you made some changes. You are asking them to pull the change in their repo (hence the term pull request). In this context, GerritHub will collect all the pull requests you will generate for the files that you enter and distribute it to your reviewers.

[I am expecting that the above parts are done at the start of the weekend. All the projects are ready. Note that you have not touched GerritHub yet.]

[At this point, you will communicate on canvas to form your teams. It is critical that you form your teams early so that you can start on Monday. DO NOT PROCRASTINATE.]

- g) Use discussion board on canvas to sign up to 7 projects as reviewers. Share your Github user id with your project owners.
- h) As project owners, find your 7 reviewers. Collect their GitHub user ids.

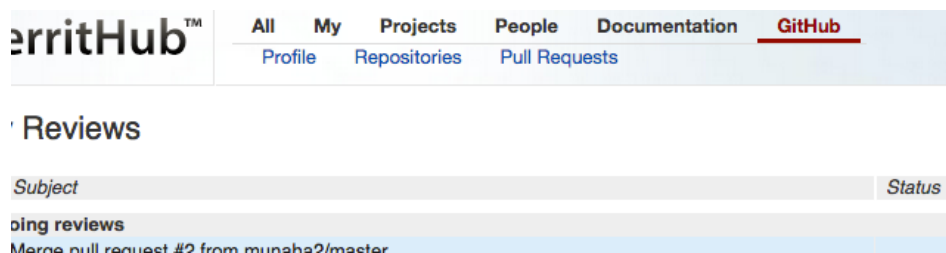
[Finish this over the weekend. On Monday, all projects start. Follow the next step.]

- i) At this point, you as a project owner have the information of your 7 reviewers.
- j) Sign in to GerritHub (<http://gerrithub.io/>). Use the first time sign in option. It will take you to Github login window. After you enter your credentials, it will take you to GerritHub again. It will first take you to a page to import your repos. Then the next window will import all of your pull requests. Then you go to the main window of GerritHub. At this point, you do not need to use GitHub any more. GerritHub will be your front end. You may follow parts of this video that explains the import process.

<https://www.youtube.com/watch?v=jeWTvDad6VM>

The video also shows some command line entries. you do not need to do them, since you will be responding to code review only (small changes to code), not using gerrit as your git repository and write more code.

- k) The video also explains the rest of the process. Watch that. However, it may be pretty confusing at this point.
- l) If for some reasons, a project owner has to import his/her projects or pull requests again, he/she can follow the Github tab.



- m) Go to see your pull requests (you are the project owner). They will be under My-> Changes part, listed under incoming reviews. They will be in bold, as in they need attention.

GerritHub™

[All](#) **[My](#)** [Projects](#) [People](#) [Documentation](#) [GitHub](#)

[Changes](#) [Drafts](#) [Draft Comments](#) [Watched Changes](#) [Starred Chan](#)

My Reviews

Subject	Status
Outgoing reviews	
★ Merge pull request #2 from munaha2/master	
★ made more changes as second collaborator - munaha2	
★ change as munaha2	
★ Update README.md	Merge C
★ MH2 - Line 2	Merge C
★ Create a.java	
Incoming reviews	
★ Update file2.txt	
★ Update README.md	Merge C
Recently closed	
★ MH2 - LINE 3	Merged
★ MH2 - Line 2	Merged

- n) Click on the text of one of the incoming reviews. It will take you to the review window. It is showing for a lot of information. Familiarize yourself with this window first. You will have to spend many many hours in your life (as a developer) on a window like this.
- o) Add reviewers by entering their login informations that you already have. As you add them, they will get notification emails.

[Reply...](#)

Owner Munawar

Reviewers

Project munaha2 ×

Branch munahaf/stockmarket2

Topic GitHub #1

Uploaded 2 hours ago

[Cherry Pick](#) [Follow-Up](#)

Related

[Add...](#)

[Update file2.txt](#)

[new file a](#)

[At this point, you (the project owner) wait to hear from your reviewers. Maybe you have started to get similar requests as reviewers from other project owners too.]

- p) As a reviewer, perform the code review, then give a ranking and pass to the project owner to fix.
- q) As a project owner, fix code to meet the review requests, create a patch and send for review again. This may happen a few rounds, until your reviewers are completely successful and have all provided you good review.
- r) When you are approved, a submit button will appear (to the project owner). When you submit, your modified + reviewed + verified code will be pushed to GitHub.
- s) Sometimes, a reviewer may not like your code change and give you negative reviews. At some point, you may want to abandon the change (there is a button). When that happens, you have to start making changes again to satisfy your reviewers.
- t) Put the info on the Excel sheet. You will put info about the 49 reviews you got in the Project Owner tab. You will put info about the 49 reviews you gave in the Reviewer tab.
- u) With your final submission, you have to take a pdf printout of the code change page on GerritHub (My->Changes).

I promise you, this will be great fun. Possibly one of the best things you will learn from your school that will help you for a long time to come (until reviews become out of favor).

Maybe this video will also be helpful. This does not describe GerritHub, but it covers gerrit in general. A bit messy, but may be helpful.

<https://www.youtube.com/watch?v=Wxx8XndqZ7A>

Deliverable:

1. The Excel File.
 2. The Modified Project: Test Cases and Stylistic/Logical Issues Fixed.
 3. The pdf file of the GerritHub changes page.
-