

Assignment 4

Version Control, Reflection, and Code Quality

Assignment Due: Friday 30th October, 2020 @ 5pm

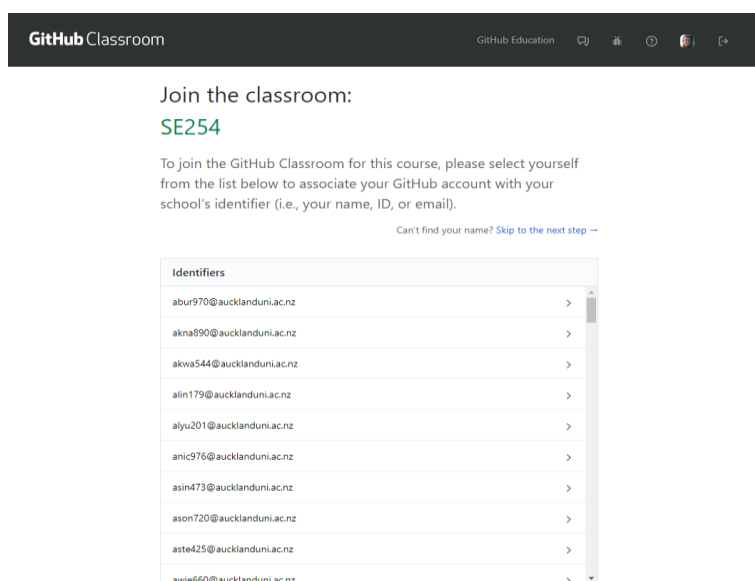
- This assignment is worth **7.5%** of your overall mark.
- Submissions must be made to *GitHub* by following the instructions embedded in this document. Failure to follow these instructions may result in your assignment not being marked!

Getting Started - Setting up Git and Github

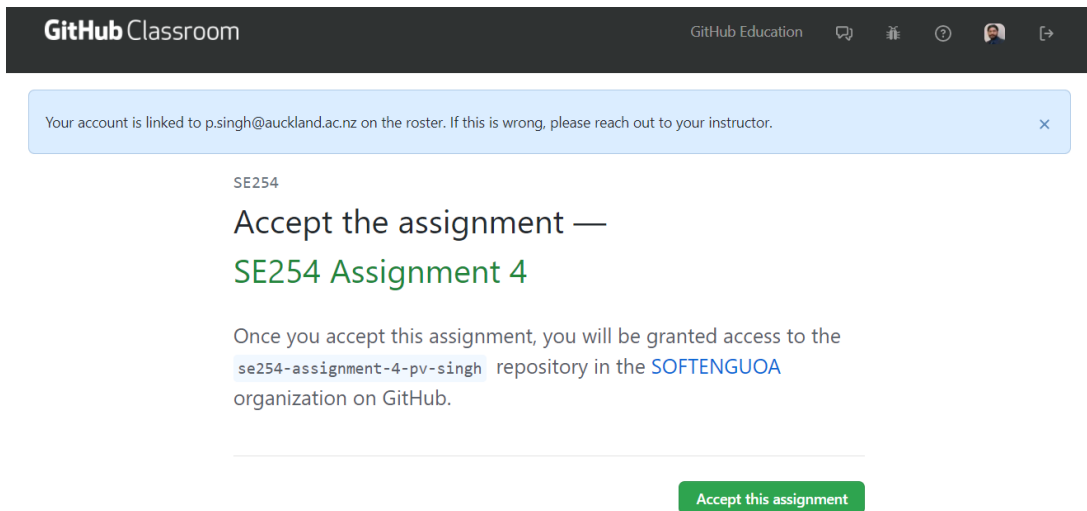
This assignment requires you to clone a Git repository from Github classroom into your IDE and perform **Task 1 to Task 3** on the same repository.

Follow the steps given below:

- Install **Git** from <https://git-scm.com/downloads>, and make sure you are able to run the basic git commands from Git *bash* or *cmd*.
- Connect **Eclipse** to Git and GitHub using the **eGit** plugin as explained in Lab 4 description.
- Go to <https://classroom.github.com/a/soyQCKmQ>



- Pick your university email address from the list to connect your Github and university email accounts.



The screenshot shows the GitHub Classroom interface. At the top, there's a dark header with 'GitHub Classroom' on the left and 'GitHub Education' and some icons on the right. Below the header is a light blue notification bar stating: 'Your account is linked to p.singh@auckland.ac.nz on the roster. If this is wrong, please reach out to your instructor.' Below this, the text 'SE254' is followed by 'Accept the assignment —' and 'SE254 Assignment 4' in green. A paragraph explains: 'Once you accept this assignment, you will be granted access to the se254-assignment-4-pv-singh repository in the SOFTENGUOA organization on GitHub.' At the bottom right, there is a green button labeled 'Accept this assignment'.

- Click “Accept this assignment”.
- There will be a screen saying that it is creating your repository. Refresh the page to receive your repository link.
- The link to your repository is of the format “https://github.com/SOFTENGUOA/se254-assignment-4-YOUR_GITHUB_USERNAME”
e.g. “https://github.com/SOFTENGUOA/se254-assignment-4-pv-singh”
- Clone the repository to your IDE and start working on your assignment tasks.
- For each of the tasks described further down in this document, you must create a new branch (and name it as listed below) for this repo from your IDE.
 - Task 1: “T1Branch”
 - Task 2: “T2Branch”
 - Task 3: “T3Branch”
- You **must finally merge** these three branches to the remote master branch **from GitHub**. See the *General* and *Task-Specific Instructions* to learn more.
- Follow on with the next tasks.

Task 1: Java Reflection (30%)

This task requires you to use the concept of reflection as studied in this course to extract metadata for any given class. All code changes and relevant commits must be performed on a *new branch*.

- Use the **Java Reflection API** library to write a program that asks the user for a *class name*, loads that class, and creates an instance of it. We assume that the class has a constructor without any parameters.
- Then, the program prints out the names and values of the public fields of the created object, and also a list of the public methods that do not have any parameters.
- The program should let the user choose a method and execute that method on the created object.
- Afterwards, the program should again show the public variables with their values and allow the user to choose a method, and so on. You **MUST** use the following class (already provided in the repo) to test your implementation. You **MUST NOT** change this class in any way **for Task 1 only**.

```
public class Counter {
    public int _ctr;
    public int _multiplier;

    //method increments _ctr
    public void increment() {
        _ctr--; //faulty increment
    }

    //method decrements _ctr
    public void decrement() {
        _ctr++; //faulty decrement
    }

    //method resets _ctr
    public void reset() {
        _ctr = 1; //faulty reset, _ctr should rather be 0
    }

    //method multiplies _ctr by a number
    public void multiplyBy(int multiplier) {
        _multiplier = multiplier; //no fault here
        _ctr = _ctr * _multiplier; //no fault here
    }
}
```

- You must implement the functionality described above by writing code for class **Reflector** and class **T1Main**, skeleton code for which is provided in the repo. *Read the inline/Javadoc comments in the skeleton code for further instructions.*
- **Stage, commit** and **push** your changes (you **MUST** write a meaningful commit message to explain the changes committed) on the current branch.
- Task 1 must have at least **one** commit (representing the task completion).

Additional Instructions

- Users should be able to run the program by running the provided **se254.a4.t1.T1Main** class.
- Note the markers may also test this program by passing any other classes that they like.

Task 2: Adding Tests to your Repo (30%)

This task requires you to make further changes to your updated repo by adding unit tests for class **Counter**. All code changes and relevant commits must be performed on a *new branch*.

Follow the steps given below:

- Add the required *JUnit* tests to the class **se254.a4.t2.CounterTest** provided to you in the repo. These tests are meant to test all 4 methods defined in the class **Counter**.
- Your tests must be meaningful, have appropriate descriptions/rationales (in the test class' Javadoc comments). The first three tests **must fail** (the failures must be because of the known faults indicated in code comments). There is no such requirement for test 4.
- **Stage, commit and push** your changes (must write a meaningful commit message to explain the changes committed)
- Fix the faults in the class **Counter** as detected by your unit tests.
- **Stage, commit and push** your changes (must write a meaningful commit message to explain the changes committed)
- Task 2 must have at least **two** commits (representing the task completion)

Additional Instructions

- Users should be able to run the program by running the provided **se254.a4.t2.CounterTest** class as a JUnit test case.

Task 3: Code Smells and Refactoring (40%)

This task requires you to work with a GitHub repository, and detect and refactor code smells in the repository code. All code changes and relevant commits must be performed on a *new branch*.

- You will use the package **se254.a4.t3** for this task.
- Identify at least one instance of any two of the following code smells:
 - Duplicate code
 - Broken Modularization
 - Long method
 - Large class
 - Feature envy
 - Unnecessary hierarchy
- Apply appropriate (one of the following) refactorings to remove the two detected code smells.
 - Move method
 - Extract method
 - Rename method
 - etc.
- **Stage, commit** and **push** your changes (you must write a meaningful commit message to explain the changes committed)
- Task 3 must have at least **one** commit (representing the task completion).

Additional Instructions

- Users should be able to run the program by running the provided **se254.a4.t3.T3Main** class.
- A successful refactoring of code must not change the runtime behaviour of **se254.a4.t3.T3Main** class, i.e. the output of the execution of the **main()** method must remain same.
- You must not change **T3Main** class. *Treat it as the “client code”*. However, you may add more classes, methods and fields in the package **se254.a4.t3**.

General Instructions – Applicable to All Tasks

In addition to the task specific instructions provided for each task, here are some general instructions applicable to all tasks:

- You **must** commit the final changes you make to your repo. You however may also commit prematurely but your committed versions **must not** have any compilation errors.
- The code changes and commits for a given task must be made on a new branch. So each task must have its dedicated branch (e.g. “*T1Branch*”, “*T2Branch*”, “*T3Branch*”)
- After completing each task, **push your new branch (directly) to the remote repo, without merging** your branches to your local master branch. Then generate *pull requests* for each branch in your GitHub account repo and accept the pull requests.
- The markers would need to see **all three individual branches representing each of the tasks (along with all commits performed on them) merged into the master** on your Github private repo. It would be **your responsibility** to make that information visible from your Github repository.

Commit Message Format

You commit messages must contain the following information:

- Type of commit (*all types need not be used for this assignment*)
 - **feat** - a new feature
 - **fix** - a bug fix
 - **docs** - changes in documentation
 - **style** - everything related to styling
 - **refactor** - code changes that neither fixes a bug or adds a feature
 - **test** - everything related to testing
 - **chore** - updating build tasks, package manager configs, etc.
- Subject or Title of Commit
 - Short description (generally a one-liner)
- Description or Body
 - Explains your changes that you made after the last commit

Assessment

Your work will be assessed on the following:

- successful/correct completion of functionality desired for each of the three tasks;
- how well you follow the instructions on:
 - creating branches
 - performing various git operations (commit, push, etc.)
 - writing your commit messages for each commit
 - writing Javadoc and inline comments

Marks Distribution

- *Task 1 (30 marks)*
 - Reflection: **Reflector** functionality (15 marks)
 - Version control: Git/Github operations, branching and repo update (10 marks)
 - Commenting: Javadoc comments and commit messages (5 marks)
- *Task 2 (30 marks)*
 - JUnit tests: **CounterTest** unit tests (10 marks)
 - Bug Fixing: Class **Counter** fixes (5 marks)
 - Version control: Git/Github operations, branching and repo update (10 marks)
 - Commenting: Javadoc comments and commit messages (5 marks)
- *Task 3 (40 marks)*
 - Code smells: Detection of two code smells (10 marks)
 - Refactoring: Application of code refactorings (15 marks)
 - Version control: Git/Github operations, branching and repo update (10 marks)
 - Commenting: Javadoc comments and commit messages (5 marks)

Related Resources and Assistance

- Week 9 and 10 Lecture resources
- Week 10 Lab Resources
- For any assistance (e.g. around assignment setup, task instructions, etc.), please email one of the teaching teams or bring your issues to the Week 11 labs.