Assignment 0

nents

Assignment 0

Assignment 0 will be due at your first discussion section (Sep 10th~Sep 12th), during your regularly scheduled section time.

Part 1: GitLab

This semester (starting with Assignment 1), we want you to submit to **Gitlab**.

Here's a quick summary:

Q. Where should I submit my assignments?

All assignment submissions must be made on Gitlab

(Try this link it the link above doesn't work: https://gitlab.engr.illinois.edu/)

Q. How should I go about using Gitlab?

1. Make sure you make a private repo for your assignment.
2. Give "Reporter Access" to **ALL TAs and your section Moderator**. All the usernames are just our Net IDs which are available on the course Wiki.

Q. Why are we using Gitlab instead of Github?

This is necessary because we still have a pilot program to work with the UIUC enterprise version of Github while Gitlab has been there for a while. The accounts on Gitlab are associated with your Net ID, which helps with any legal issues in the future.

Part 2: Code Smells and Refactoring

This is based on the Refactoring Lab Session exercise developed at LORE, and modified by @Chen, Nicholas Chun Y and @Lee, Yun Young. You might notice some peculiarities with the code because I suspect that the group who designed this consisted mostly of C++ programmers (they also released a C++ version of the same exercise). Therefore their code doesn't really follow the Java convention. You may change it if you want to, but that is not the main purpose of this homework.

For discussion sections, you must present your code using an IDE (integrated development environment). We will run through this assignment using both Eclipse and IntelliJ IDEA.

Summary

Table of Contents

Reading

- Style Conventions
- Modular Programming
- Code Smells

Submission

This assignment is due at the beginning of your discussion section the week of September 10, 2018. Please be sure to submit in GitLab, and ask your moderator or TA before the deadline if you have any questions. Also, make sure to place your work in a folder called Assignment0, matching this spelling exactly.

Objectives

- Learn about code smells, coding styles, and refactoring
- Learn to critique code

Resources

- Code Smells
- Refactoring

Grading

Category	Weight	Scoring	Requirement
Participation	1	0-1	interact with the group 2 times (ask a question, make a comment, help answer a question, etc.)

You can download Eclipse for free right here:
https://www.eclipse.org/downloads/

Category	Weight	Scoring	Requirement
			400 lines and check in the code written for this assignment to our Gitlab repo
Attended	1	0-1	attended the discussion on time
Total	**3**		

nents

Few of the course staff, are IntelliJ fanatics. There is a free community edition that can do Java, Android, etc.

Context

You are facing a software system which represents a simulation of a local area network. The development team has been very fast in accommodating the initial requirements for the system and has been able to release version 1.4 of the system, which contains all the functionality for the first milestone. However, the customer now requests for the remaining functionalities and the development team fears that the current design is not up to the task. Having heard of your refactoring expertise, they hired you to have a look at their code and refactor it appropriately. They do not expect a perfect design, yet they want to be able to add the remaining functionalities easily. They told you that they have regression tests available.

Getting Started

First, read the required reading for this assignment, paying special attention to code smells. You can complete this assignment using either Eclipse or IntelliJ.

Using Eclipse
1. Download the HW0.zip file.
2. In Eclipse, select File -> Import.
3. In the window that pops-up, expand the "General" folder and select "Existing Projects into Workspace".
4. Click on the "Select archive file" radio button.
5. Select the location of the HW0.zip file that you have just downloaded.
6. At this stage, the project should have been imported and an unzipped version has been created in your workspace folder.
7. Check it into your Gitlab repository

Using IntelliJ
1. Download the HW0.zip file.
2. In IntelliJ, select "Create new Project", name your temporary project, and click through the prompts with the default options.
3. Extract HW0.zip and copy its contents into your root project directory.
4. Check it into your Gitlab repository

Read all the Code in 5 minutes

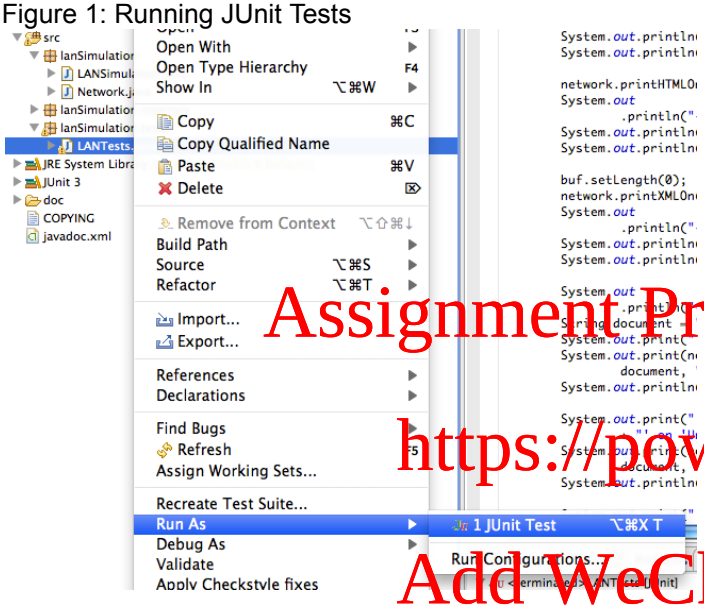
From within your IDE, read the code. Use the features of the IDE to help you navigate the code quickly. For instance, if you are in the middle of a portion of code and want to see where a class/method is defined, hold on ctrl (or the apple key) and click on the word.
`LANSimulation.java` is the driver class of the system.

While it may help you to understand the code, its functionality is not important for this homework. It is included for testing purposes.

nents

`LANTests.java` contains regression tests (*can you explain what regression testing is?*). In Eclipse, run the tests by selecting `LANTests.java` in the Package Explorer, click the right mouse button and select Run As -> JUnit Test (#Figure 1). In IntelliJ, open `LANTests.java`, click on the class name `TestCase` and click "Add junit.jar to classpath", then right-click and hit "Run 'LANTests'". After each refactoring, you should run the tests to make sure that refactoring didn't introduce any bugs.

Figure 1: Running JUnit Tests



Issues with JUnit
You may face issues with your IDE identifying the JUnit library. We recommend downloading Junit 3.8.1 from here. This seems to have fixed all issues with IntelliJ. You can see a GIF walkthrough of the whole process here :

http://imgur.com/QL2z3Pi

Further, we also tried the following steps on Eclipse and fixed the issue.

1. Download Junit 3.8.1 from here.
2. Right click on `project` in Eclipse and go to properties.
3. Go to "Java Build Path" on the left hand side.
4. Click on the "Libraries" tab.
5. Click on "Add ExternalJARS..." button and choose the Junit 3.8.1 jar file you downloaded.
6. Click on "Apply" then "OK"

7. Now try run "LANTests.java" as JUnit Test

ıents

This section runs through instructions for using Eclipse to perform several automated refactorings, but the process in IntelliJ is nearly identical.

a. Extract Method

Have a look at the `printDocument` method in `Network` class. Several calls to `report.write` method are duplicated (*what's the name of this code smell?*). In order to remove the duplication, we need to perform the Extract Method refactoring. However, you may have noticed that not all the statements are exactly identical:

```
report.write(">>> Postscript job
delivered.\n\n");
. . .
report.write(">>> ASCII Print job
delivered.\n\n");
```

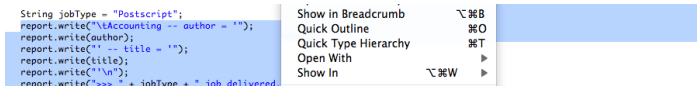
You should first perform Extract Local Variable by hand (#Figure 2). Now all the calls to `report.write` method are identical! In order to perform Extract Method refactoring, select the statements that are repeated and select Refactor -> Extract Method... from the right mouse button menu (#Figure 3). You can also use the Refactor menu in the toolbar. Then give a meaningful name for the new method (#Figure 4).

Figure 2: Perform Extract Local Variable refactoring by hand so that your code looks like this

```
        String jobType = "Postscript";
        report.write("\tAccounting -- author = '");
        report.write(author);
        report.write("' -- title = '");
        report.write(title);
        report.write("'\n");
        report.write(">>> " + jobType + " job delivered.\n\n");
        report.flush();
    } else {
        title = "ASCII DOCUMENT";
        if (document.message.length() >= 16) {
            author = document.message.substring(8, 16);
        }

        String jobType = "ASCII Print";
        report.write("\tAccounting -- author = '");
        report.write(author);
        report.write("' -- title = '");
        report.write(title);
        report.write("'\n");
        report.write(">>> " + jobType + " job delivered.\n\n");
        report.flush();
    }
```

Figure 3: Invoke Extract Method refactoring



nents

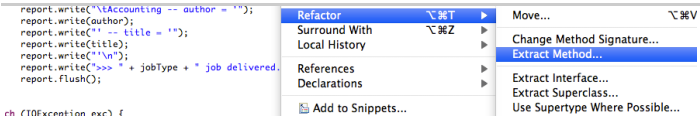
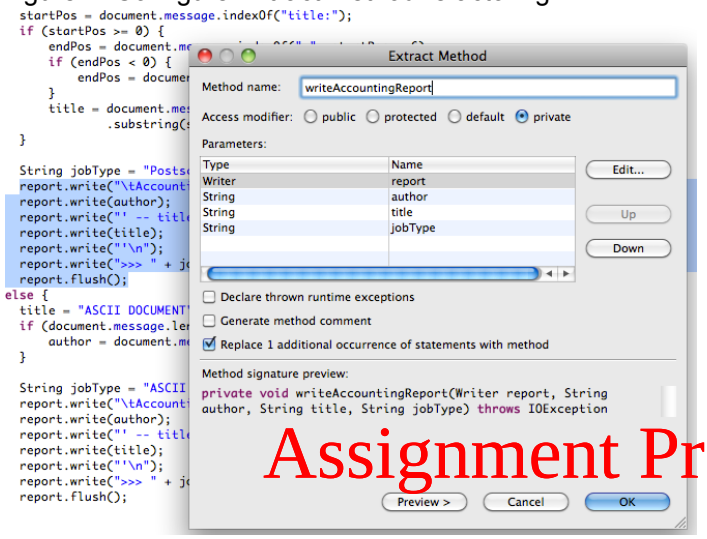


Figure 4: Configure Extract Method refactoring



Now it's your turn - look at the `requestBroadcast` and `requestWorkstationPrintsDocument` methods in Network class. There are calls to `report.write` method that are repeated once inside `requestBroadcast` method and twice inside `requestWorkstationPrintsDocument` method (starting with `report.write("\tNode ");`). Get rid of the duplications by applying Extract Method refactoring. Note that the three clones are not exactly the same so you'll first have to modify the code a bit before doing the refactoring.

b. Move Behavior Close to the Data

Having extracted the above methods, you note that none of them is referring to attributes defined in `Network` class, the class these methods are defined upon. On the other hand, these methods do access public fields from the class `Node` and `Packet` (*what's the name of this code smell?*).

The `printDocument` method accesses attributes from two faraway classes, yet does not access its own attributes. You should move this method to the `Node` class, more precisely the `printer` instance of the Node class. Select the `printDocument` method name and invoke Move refactoring (#Figure 5). Select the printer instance and continue with the refactoring (#Figure 6). (*Do you understand the warning messages?*)

Figure 5: Invoke Move refactoring

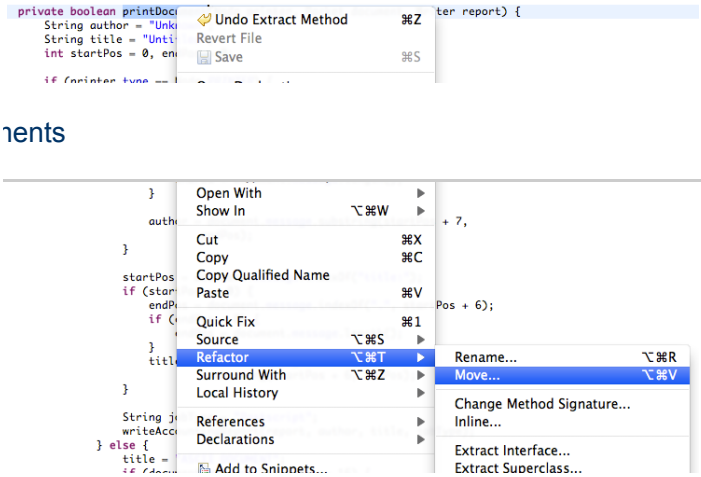
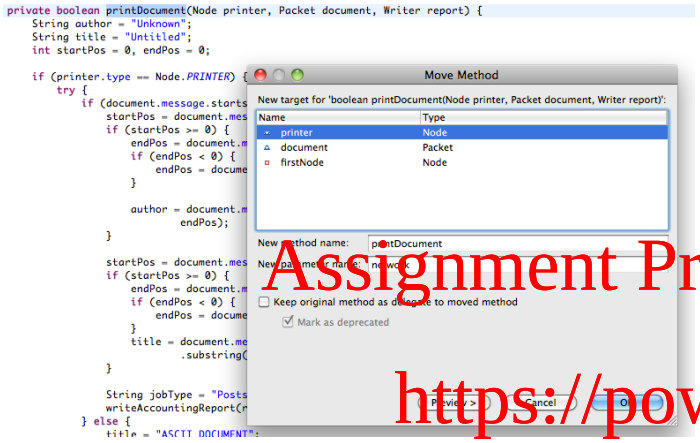


Figure 6: Configure Move refactoring



Now it's your turn - The method you just extracted from the `requestBroadcast` and `requestWorkstationPrintsDocument` methods does not belong in `Network` because most of the data it accesses belongs in another class (*which class would you choose and why?*). Apply Move Method to define the behavior closer to the data it operates on.

Part 3: Bring Your Code

Bring in *any* piece of code that you have written that is at least *400 lines* long. You must check this code into Gitlab. You will also be asked to ask questions and discuss the code presented by other students in your section, so please read Style Conventions, Modular Programming, and Code Smells.

If you bring your own laptop with you to your discussion section, please make sure ahead of time that it can connect to an external display properly and bring any necessary adapters if you have a laptop such as a Mac (our displays have VGA cables). It is your responsibility to ensure that you are able to present your code each week.

The only stipulation for this assignment is that the code you bring in **must be the code you have *written yourself***. You will be asked to present it to the group, and answer questions about it. For these reasons, it is best if you pick a recent work. In addition, it makes the discussion

more lively if you bring in something other than an MP from a previous course, but you don't have to write something new if all you have is MPs.

nents

markup languages, or assembly code.

You will be asked to walk us through your code, and identify as many code smells and possible refactorings that can remove the code smells as possible. If you notice code smells in other students' code, you should point them out and discuss.

Lastly, you **must** present your code, regardless of language, using an IDE, ideally Eclipse or IntelliJ IDEA. This is so that you become familiar with Eclipse or IntelliJ IDEA since we will be requiring you to use it for the first few assignments. There are plugins for Eclipse and IntelliJ available for most languages out there, so you should be able to customize the IDE to demo your code.

无标签