**Swap 2 HW**

**CSSE375-02&03, Spring 2014**

This is the **second** of **multiple “swap” activities** among small teams, doing coding as a HW activity.

Proposed changes in red, *below*.

The learning goals are as follows:

1. Considering strategic system goals in deciding what to refactor.
2. Evaluating the “quality” of peers’ code, in these ways.
3. Using their recommendations in adding new features. (And *supplying* recommendations to the team you gave your code to.)
4. Applying more of the specific coding skills we’re learning in 375, about refactoring.

This time, you’ll be adding features and fixing bugs to a system that came from Swap 1.

At the end of **this week**, again, you will pass your code to another team. They will then do the following:

1. Grade your code immediately, based on the rubrics of that particular code swap.
2. Add to and bug fix your code, making it “their code” for the next week. Bug fixing should be focused on major bugs, those which can’t be ignored. See comments in red for point B in the rubric table.

* **Repeat Warning:** We’ll do the “exchange of code” during each Monday’s class, starting next Monday, so it really MUST be done and running by then!

**What’s it supposed to do this week?**

What will we do during the second week of this Swap HW assignment?

I. “Today,” on Monday, Feb 24, do the following in class:

1. **Exchange 1**: See the revised “Teams for everything - Projects and Swaps!” Spreadsheet on Moodle. Notice that you will be emailing a zip file to one team (send it to both people – or all 3), and receiving one from a different team!   
     
   Note: Do not zap every copy of the code you are sending to another team. You may well want to see it, later on! (E.g., if they send you questions, in today’s grading.)
2. Also during class today, use the Swap 1 Code Dropbox to turn in, to me and the TA’s, the same zip file.
3. Revise your code-sharing system so that you are now sharing the new program you received.
4. Bring your laptops: Both partners on a team need to get the program running during class today, and verify what functions you believe it does or doesn’t do. If the team who worked on it changed something, and it just works in a different way, this gives you a chance to clarify that with them.
5. Start grading it, to see how that feels. Use the rubric for Swap 1. Attach the bug list below it, if there are bugs. **This is what you will turn in to the Swap 1 dropbox by 11:55 tonight.** Be sure you graded everything worth points, so that it adds up to as many points as the other team should get!
6. Ask questions about the enhancements (“features”) they recommended but didn’t add (see *below*).

II. During the week, do the following things with your partner, to your copy of this new code, verifying after each activity that the system still runs:

1. On every coding change or comment you make, **precede that line or area of code with a comment line** which says: // SWAP 2, TEAM XX where XX is your team number. This way, it’s easy for the next team to find all the changes you made, and use the rubric to grade those.
2. **In Swap 1, the other team was asked to find as many additional code smells as possible.** Their goal for perfection on this was to find a total of 10 of the 22 **different** smells that Fowler identifies, one non-trivial example of each. You’ve already looked at these, because you just graded them.  
     
   They were also told in Swap 1, “On each SMELL you identify, also **describe the kind of enhancement** that the change would enable. **Don’t do those changes**, but do comment them in enough detail that someone else could do so.”  
     
   Great – Now, for Swap 2, your job is to take them up on those recommendations. Whatever were the 10 or so enhancements they thought some refactoring would enable, do just the refactoring for **7 of these**. Pick the **best** 7, so you don’t have to add in the comments that one of them was an unworkable feature, and all you can show is a best try at it. What are the best 7? Those that appear to meet some strategic system goals for this system, and what it’s supposed to do. The more important the feature is, to the success of the system, to more reason you should do it. So, you may want to start off listing all 10 smells they found, and the features they thought would be enabled by fixing them, and decide what is important.  
     
   On each, put a comment line like, // REFACTORING FOR ENHANCEMENT FROM BAD SMELL. Then, in your comments go on to describe the following:
   1. What refactoring you used as the first step. This time use one of Fowler’s, unless you can explain why none of his fit this.
   2. In comments, clarify what you believe could be added as a feature, as a second step. Goal – do it without now changing the existing code, just adding to it.
   3. Your analysis of how successful the refactoring was!
3. In Swap 1, the team working on this were asked to “Make the program do an **additional feature** of your own invention, something which is difficult to add with one of the bad smells you already identified above, but easy if you fix that smell! Label this one as ADDITIONAL FEATURE, and make your comments that go with it clear, so that the team receiving your code in the next Swap can check it out.”  
     
   Ok, just for fun then, do an **elaboration** of their feature, which requires you to refactor it first. Because they picked something unique to do, you may have to think about this for a bit. Label this one // FURTHER ELABORATION. If for any reason the other team didn’t do this, or you feel they did something that’s impossible to enhance, let me know by our lab time on Wed, so I can look at it, too.

Let me know if you have questions!

**The “turnin” for this assignment**

1. As with Swap 1, this will be “due” at the start of class, the following Monday, in this case, Mar 31. That’s a hard deadline, because if you miss it, you leave the next team, who will work on your code, with nothing to do! The turn-in will be a zipped version of the Eclipse program files for the program, like the way you received it, but now ready to pass on to the next team.
2. In class, you will find out the names for the team you will send it to. You can email the zipped files to them directly, at the start of class.
3. During class, you also need to turn in the zipped files to me, in a drop box on Moodle! One copy per team.

**The grading**

Here’s the rubric to use in grading the system you receive. You can start grading during class on Monday, March 24. You will need to finish grading, and turn-in your rubric form as a .docx file, in the Moodle drop box for this, by 11:55 pm next Monday night.

SWAP 2 RUBRIC:

CSSE 375 Section # \_\_\_\_ From Team # \_\_\_\_ , Graded by Team # \_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Capability / Feature / Assignment** | **Grade (“works perfectly” or “done perfectly” = full credit in each case)** | **Grader Comments – Explain anything that is missing, doesn’t work, or has issues or concerns.** | **Rubric Comments and Examples** |
| **0. It still runs** – Does all the “Required” and “Additional” features for the original CSSE221 assignment, that it did before the team changed it. | \_\_\_\_ out of -30 |  | This time, not running is a negative score. Take off for things that don’t run. If they don’t have anything to give you next Monday, don’t show up in class, or the whole thing blows up when you try to run it, extend this grade to a 0 on the whole assignment. |
| 0’. They made wholesale changes to the system that were not asked for. | \_\_\_\_ out of -100 |  | This is a negative grade if they did it. **Zero** if they avoided the temptation to do this. -100 if they rewrote the whole program without commenting where they made changes. -30 if they changed key data structures, like how the calendar is represented. |
| 0’’. They didn’t follow the documentation guidelines to make it easy for you to grade this. | \_\_\_\_ out of -20 |  | Also a negative grade. Take off some points if they didn’t put the SWAP 1… comment lines in, or the more specific ones asked for, like SMELL. |
| B. They fixed bugs identified from grading, and refactored bad smells, and they seriously tried to clarify the possible enhancement to add to the system, recommended by the Swap 1 team. A total of 7 fixes or refactorings were made. | \_\_\_\_ out of 10 for each of the 7 asked for (70 total) | Please make a list, so they can see clearly what you took off for. | Grade proportionally if you believe they could have tried harder to refactor or fix this, or to describe how to implement a good feature based on this smell and refactoring. Did they pick features that accomplish **strategic system goals**? Note: In this case, duplicate refactoring types are **ok**, but they should not be trivial. Note that the bug fixes should be ones needed to run the system (e.g., major), and should be few. (They should not have replaced the idea of doing refactoring in this homework with an assignment to do all possible bug fixes.) |
| C. They added to the other team’s “one additional feature of their own invention,” refactoring first. It is well described, well written, and works. | \_\_\_\_ out of 30 |  | Same as *above,* but also they should have actually implemented something, and they should show that they used the two-step process, so that changes were done as refactorings, then code was added to do something new. |
| Total | \_\_\_\_ out of 100 |  |  |

Below the table, in the .docx that you hand in, note additional bugs found that need to be fixed:

1. Description:
2. Etc.

Please observe that I also plan to grade you on the quality of your grading, based on my own review of the same code!

**The next cycle, Swap 3!**

After doing your grading of what you receive, you can start on the next cycle of changes. You will have a new assignment with new things to do, like more refactoring, for Swap 3 – and some surprises. I’ll be writing those as you’re doing this assignment!