**Refactoring Process Notes**

***Disclaimer: Just my chicken scratching of notes.***

***Assignment by: Jeff Moreau***

1. I opened the assignment and studied the screen shots, code, and instructions. I first noticed it’s not written in Java. It appeared to be C# so I went and confirmed that.
2. C# and Java are very close, but I reviewed an online tutorial to brush up on syntax as I never worked with C# before.
3. I created a GIT project and pulled the file up in VSC.
4. The code appears as one large method taken from a class. I went ahead and wrapped this method in a class. I wanted to visually see it as a class and for formatting purposes.
5. I started by looking for obvious code redundancies. I removed that portion of code into another method and cleaned it up. For all places of redundant code I replaced it with a call to that method.
6. I refactored like this a trillion times.
7. Once obvious redundancies are pushed off to methods, I reviewed for logical groupings of business functions within the code. The original main method is a method to process the call notes. Inside this method we create and populate the CallLogUI object. The method than calls the DBLayer class/method to save the data. I wanted to pull out the creation of the CallLogUi so I copied all of that code and put it inside its own method. This way other methods could call and create the CallLogUI object.
8. I refactored some code that is required in both the createCallLogUI and the InsertUserCallNotes methods.
9. I now created a main method. This would not be here but I wanted to indicate where the starting point was.
10. I removed any unnecessary comments regarding when code was added. This information can be obtained from a code repository.
11. I added more useful comments. I decided use a lot of comments given that there was a lot of code present, a lot of names were abbreviated, and things being saved in different orders.
12. I added method header comments. To distinguish better where the methods were, I was going to use a lot of “\*\*\*\*” astericks as markers. Probably not the best practice, but helped in this assignment.
13. I then reviewed the setting of fields for the CallLogUI. I moved up fields being set where it didn’t use any values from above it. I wanted to group the setters together to help the developer identify them quicker rather than looking everywhere.
14. I did add some TODO comments on items I would go back to the Project Manager/Business with. I would like more clarity on the business functionality so I know where lines of code should be where they are. Some of the items I would look for clarification on were the same field was being set in one line and then set again a line or two down. It seems like we could reconfigure that so we are not unnecessarily setting the fields. In addition, it’s hard to identify what’s going on. I did leave the placement of the code as is because there might be a reason for the code placement.
15. Once I completed the refactoring. I went back to the original file to review each line/section with my refactored copy, just to make sure my changes performed the same way as the original.
16. For a real world project, I would highly suggest doing a ton of JUnits and side by side comparison tests between environments to make sure the code is acting the same.