SubHunter Refactored

Submit Assignment

Due Wednesday by 5pm **Points** 50 **Submitting** a file upload
File Types java, png, and jpg **Available** Aug 30 at 8am - Sep 11 at 11pm 13 days

This exercise builds on the first exercise using SubHunter. The Sub Hunter game described in the Pakt book discussed in class can be found at the following URL:

https://github.com/PacktPublishing/Learning-Java-by-Building-Android-Games-Second-Edition/tree/master/Chapter07 **(https://github.com/PacktPublishing/Learning-Java-by-Building-Android-Games-Second-Edition/tree/master/Chapter07)** **(https://github.com/PacktPublishing/Learning-Java-by-Building-Android-Games-Second-Edition/tree/master/Chapter07)**

(https://github.com/PacktPublishing/Learning-Java-by-Building-Android-Games-Second-Edition/tree/master/Chapter07) After you have successfully compiled and run the application, modify it as follows:

1. Abstract the submarine into its own class **Sub**. You make whatever decisions that you feel are necessary with respect to object oriented practices.

2. Abstract the grid into its own class **Grid.** You make whatever decisions that you feel are necessary with respect to object oriented practices.

3. Create whatever other classes you feel are necessary for the application.
   1. Note, I expect that you will think extensively on this point.
   2. Your assignment will be judged primarily on how well you managed to incorporate the material presented in class.

4. Modify the game as follows:
   1. Your game will place three subs on the grid, their locations must be distinct.
   2. After each shot the game will report only the distance to the closest sub.
   3. After each of the first two subs are hit, the game will mark the sunk subs with a red grid square.
   4. After all three subs are sunk, the game will finish as the original game.

Think about what the function of each of the new classes should be in the context of this game. How do these classes interact with each other? There is not one fixed or correct answer to this. You will be graded on how much effort you put into thinking about this simple program in an object oriented way. Put effort into documenting your classes with ***significant comments*** that describe your thought process.

Documenting your code, for this assignment only as comments within the code itself, is a major contributor to your grade in this class. In particular, you should be discussing the reasons for your choices of decomposition and design. This project is simple, in fact, trivial. The assignment is not about can you modify a trivial game with some trivial extensions and get it working. That should be obvious to you as students. If you found yourself saying "whoa, this assignment is easy, I hope that the rest of the class is going to be cake," and you haven't provided any documentation, then you might be missing the point of the class, and certainly of this assignment.

What classes did you ultimately choose? How do they interact? How does this relate to the game itself? Why are your choices preferable to alternatives? What were the alternative choices that you rejected? Don't go crazy, but put effort into this. The assignment code provided in the Github is not at all sufficiently documented in this regard.

The more readable and self-documenting your code is, the less additional documentation you will need to provide. Let me be clear on this: how you present your work is important in this class, in other words, neatness counts, coding style counts, readability counts, documentation counts, participation within the group counts, communication counts. The value of the documentation is more than just communicating to others, it's about communicating and documenting your thoughts for yourself so that as you refactor you don't break existing ideas.

IMO, this is much easier to do as you code then to wait until the last minute to try and document in order to "get a good grade." For this assignment, ask yourself this question "am I providing sufficient commentary to communicate to a reader that I understand the basics of object oriented design." If you put a good faith effort into that, even if you aren't yet very good at OOP design then your grade will be fine. If you choose to ignore the requirement to document, or simply generate text that does not reflect a good faith effort to communicate your understanding, then your grade will suffer.

I don't want you guys losing sleep over this on this "get to know you" exercise, but I don't want you to think that you can simply gloss over the documentation requirement either.

This is an individual assignment.

Submit a single java files **SubHunter.java** Submit a screenshot of the game after you have sunk the first two subs.

Include all additional classes in **SubHunter.java**. While it's good practice to put each class in its own file, for this exercise you may use the following syntax placing your non-public classes at the top level of the SubHunter.java file.

```
class Sub {...}
class Grid {...}
class ...? {...}

public class SubHunter extends Activity{
    ....
}
```