**Chris Park Post Task 3 Comparisons**

**Comparison Myself vs Christian**

**Weaknesses:**

Perhaps first to come to notice, and glaringly so, is that my solution does not correctly adhere to the Strategy pattern. In hindsight the use of interfaces in the solution was more of an attempt to be clever than to be efficient, and was overall a misguided idea. This coupled with the fact that I don't have any overrides for the original A\_Vehicle methods makes it impossible for a Uboat to conform to the pattern correctly. Christian was cognizant of these requirements and implemented them correctly.

Christian's classes don't access member fields directly in the constructor like mine do. This makes for better error checking when initializing the object. Where as by directly setting the initial values of all of these fields in the constructor like I do bypasses all of the runtime exceptions. I also have a horrendous amount of checks against null values, which as I understand now is quite the code smell, and a habit that is hard to break. Christian deals with these situations by using the boolean has methods instead, a much better solution.

**Strengths:**

Perhaps the only claim I could make as a strength is a bit of a relative observation, and one that I'm not entirely sure is a correct one. From a stylistic standpoint, I have an aversion to if/else statements. I prefer to organize the logic so that in the event an if statement is used, it remains fairly small and doesn't chain into multiple else and else if checks. I think it helps to reduce code clutter from bracketing multiple code blocks, and helps with readability.

**Comparison Myself vs Andrey**

**Weaknesses**:

Other than stylistic differences when bracketing if statements, and the obvious error of my implementation of the solution not working correctly, the differences between our code logic are a lot less stark. Andrey's code is very readable, and easy to follow. When it comes to naming conventions, member fields of a class are preceded by an underscore in my solution, where as Andrey uses the “this” operator to communicate which variables are fields of the class. I'm not sure that one is better than the other. The this operator is certainly a more direct and clear way of clarifying the scope of the variable.

When dealing with returning boolean values in the has methods, Andrey simplifies the returns into a single boolean statement, where as I was using at least one if check. This resulted in these methods containing a fair amount of extra and unnecessary code on my part. This is a habit I intend to pick up on.

**Strengths**:

In terms of naming conventions for class files, Andrey didn't precede his abstract classes with the A\_ to clarify them as abstract. This seems to me a small thing, but useful none the less, and he commented on the potential strength of doing so during our meeting on Wednesday. So we are in agreement there.

**Comparison Andrey vs Christian**

**Strengths for Christian:**

Most of the comparisons I made between my solution and Christian's appear to apply here as well.

Christian's use of method calls instead of directly setting member fields is beneficial and makes for better error checking.

Stylistically Andrey uses more “{}”, in particular when dealing with single line if statements, where as Christian does not. I don't know that I'm qualified to make a judgment call on which approach is a better one. Andrey spoke to me about his reasoning behind doing it this way, chiefly that it was a safeguard against potential mistakes he might make; that he didn't trust himself. Given this reasoning I can see where he is coming from. That being said, excluding the brackets in single line statements following these if statements is a stylistic choice that I share with Christian.

Again, as in the comparison between my solution and Christian's, Christian makes a good call by overriding the single power plant method calls to work with the dual engine specific methods. I suspect this may be the type of foresight that Dan is looking for in a robust solution.

**Strengths for Andrey:**

As with the comparison between Andrey's solution and mine, the use of the this operator is potentially a more direct solution to naming conventions for class field variables. Christian shares my use of a preceding underscore, which stylistically I am used to, so I may be a bit biased here.