Jace Riehl
Introduction to Software Engineering
Assignment 3 Report

Extending a Developers Code

In general, it was fairly easy to implement the developers code that I received. If I needed a function from his code, I could take a guess as to what it would be called and a majority of the time the autocomplete would find it. Specific to this code, the easiest part of extending it was using the functions that had already been created. His functions did exactly what I needed them to do, they were clearly named to the point that I didn't need documentation for the most part, and he had nearly every function that I needed. Working with another developer's code in general, the easiest part was trusting that the developer knew what he was doing when implementing the functions. Being able to focus on what I was doing with cheat and not getting slowed down by having to create or check that the function was working made it easier for me to develop cheat.

Challenging aspects

The most challenging aspect of implementing another developers code was trying to understand what some of their functions were doing. An example of this would be trying to figure out why you couldn't just take cards from a player. His coding style is different and it made it hard to understand what he was doing with his sometimes.

Rating the code

The quality of the code that I was given the code I was given is a A-/B+ level code. It was all well documented, he uses exceptions to guard inputs, his functions worked well and were correctly named. The reason I would not rate it an A is because of some bad coding practices that I discovered while working with his code, but didn't notice in the code review. It is not possible to directly take a card from a Player's hand you can only give it to another Player. The result of this was him using a Player as a discard pile itself, which doesn't make sense. He also used a model class to encapsulate his model variables and it makes adjustments to the models, such as giving the Players cards. I didn't bring it up during the code review because it would be a big change for him that wouldn't impact the design of Cheat. There were no abstract classes made, showing not extensive forethought for the extension of cheat, but I was able to make it work. Overall, his code was well written and extensively tested so I didn't have many problems working with him code.

What the previous developer could have done

There are two distinct things that David could have done to make new feature development easier, both of which I touched on earlier. It would have made more sense to have a view and controller abstract or interface class. It would have been a bit easier to develop cheat if some functionality had already been implemented in the base class. Dealing cards to players is a general functionality in a card game and it would make sense to have that in the base class among a few other classes such as a prototype for running the game. The second thing that could have been done is making the Player class more general. I had to make a general function that removes cards from Player’s hands and a function that sorts their hands. Both of these functions are useful for many card games and it would be useful to include those.

What I could have done differently

I didn’t create abstract classes in my code which would have made it a bit easier for the person developing on my code a bit easier. It would have saved from having to inherit every function from my view or controller in order to extend it. I could have made some functions constant which would have been a better coding practice. Overall, I think i did a good job setting the developer up for success. My functions for the model were generalized so It was easy for them to work with.