Andreas Bom
ab22cw

Peer Review

Workshop 3

Review on Sebastian Lindblad, sl222yy

Try to compile/use the source code provided. Can you get it up and running? Is anything problematic?
There is no problem to get the application up and running.

Test the runnable version of the application in a realistic way. Note any problems/bugs.
At first run, in the configuration provided, there is no bugs or any problems.

Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction?
New implementations includes interface WinsGameRule, and two classes (StandardWinRules, VariantWinRules) implements this interface. The Soft17 strategy is implemented in BasicHitStrategy. The diagram and the implementation correlate. I can't find missing relations or relations in the wrong direction.

Is the dependency between controller and view handled? How? Good? Bad?
The dependency between the controller and view is removed by simply moving the method that takes user input, from the view to the controller. This means that the controller reads user input. The responsibility to read input should be in the view. The controller should be like a facade from the view to the model[1]. There will always be a dependency between controller and view, but the "ugly" dependency that the original code had, was that the input character followed throw from the view to the controller. (for example, input 'p' was 'Playgame'). A better way to handle this dependency would be to use an enum (could have been <<enum>> p=Menu.PlayGame).

Is the Strategy Pattern used correctly for the rule variant Soft17?
The Soft17 is implemented in the BasicHitStrategy. This is confusing, since the file name and the class is called BasicHitStrategy and the rule that is implemented is called Soft17. My recommendation is to change the file and class name, so it matches the strategy name. The implementation itself, is correct implemented and the 'rule' is working as expected.

Is the Strategy Pattern used correctly for the variations of who wins the game?
The strategy pattern is implemented in the correct way. The naming could have been better. The name is not telling what the strategy is doing. If it is difficult to give an explicit name, it would help if there is some comments in the beginning of the class, telling what the strategy is doing.

---

[1] Larman, C (2005). *Applying UML and Patterns (Third Edition)*. Upper Saddle River: Prentice-Hall. Chapter 17:13 Controller

Andreas Bom
ab22cw

Is the duplicate code removed from everywhere and put in a place that does not add any dependencies (What class already knows about cards and the deck)? Are interfaces updated to reflect the change?
The duplicated code is not removed from InternationalNewGamStrategy, AmericanNewGameStrategy, Dealer::Hit(), Dealer::Stand().

Is the Observer Pattern correctly implemented?
The observer pattern is implemented by the .NET build-in functionality 'event EventHandler'. This is out of my scope of knowledge. The only thing I have to say about that, is that it doesn't really show if the programmer understand the pattern or not.

Is the class diagram updated to reflect the changes?
You can't see that the observer pattern is implemented in the class diagram (since the pattern is implemented throw .NET functionality).

Do you think the design/implementation has passed the grade 2 criteria?
The programmer has not shown that he understands the observer pattern. He has shown that he can use the observer pattern by using the build in functionality. I my opinion, he solved the assignment in a creative way and the written assignment did not state that this way was not allowed. For that reason, I think he passes this part of the assignment.
The refactoring of duplicated code has not been done. To me this is a small part of the assignment. Overall, I think the assignment was well done and that is should pass grade 2 criteria.