

# Peer review

## **Test the runnable version of the application in a realistic way. Note any problems/bugs.**

The program runs fine either from visual studio or using the .exe file. I came across one bug that would occur when you enter "s" for stand, it made the console print the next card drawn 6 times (but only one card would be put in your hand afterwards). However this bug resolved itself for unknown reasons.

## **Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction?**

The class diagram reflects the code and all the classes and relations are there as far as i can tell.

## **Is the dependency between controller and view handled? How? Good? Bad?**

The dependency has been handled by doing the compare with the input in the view, returning the corresponding enums to the controller. The solution looks good and has successfully removed the hidden dependency between the view and controller.

## **Is the Strategy Pattern used correctly for the rule variant Soft17?**

The SoftSeventeenGameRule is implemented and implements IHitStrategy as it should. The logic of the rule correctly checks if there's a combination of ace and six in the hand and if so the dealer will take another card.

## **Is the Strategy Pattern used correctly for the variations of who wins the game?**

TieDealerWins and TiePlayerWins are correctly implemented.

## **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 found in the Dealer class has been dealt with. The solution did not require any added dependencies.

## **Is the Observer Pattern correctly implemented?**

The Observer pattern is implemented and working. You use player for subject and the controller for the observer. However i think it is recommended to make a Subject interface or

an abstract Subject class that you then extend/implement in the classes you want to act as subjects. That way it's easier to add new subjects and make changes in the code that affects what you want to observe. It is also easier for the reader of the code to understand the structure this way.

**Is the class diagram updated to reflect the changes?**

Yes, you seem to have added all the new classes and interfaces you implemented and used correct arrow notations.

**Do you think the design/implementation has passed the grade 2 criteria?**

Yes.