Peer Review

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

The jar file is executable from command prompt. The program seems to work fine and blackjack is playable and working. Nothing to add here.

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

The diagram is a bit hard to read but as far as we can tell it corresponds fairly well to the code. We found a missing dependency between the controller and model.Card in the NewCard method. This dependency should go from the Observer interface and the Card class.

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 SoftSeventeenStrategy is implemented and implementsIHitStrategy 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?

Yes the Strategy Pattern is correctly used in the same way as for the hit rules. The logic of BasicWinStrategy and PlayerWinStrategy is correct.

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, except for the one missing dependency mentioned above the diagram seems to reflect the new implementations you have done.

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