Workshop 3 Peer Review – Andreas Anemyr

Try to compile/use the source code provided. Can you get it up and running? Is anything problematic?
Program works perfectly. No issues found

Test the runnable version of the application in a realistic way. Note any problems/bugs.
When the first card is dealt to both dealer and player, the player's card is duplicated for the dealer. For example, the player is dealt the 10 of Clubs. The dealer then appears to be dealt the 10 of clubs. Then the player is dealt a second card at which time the dealer is dealt a second card and the 10 of clubs changes to whatever the dealer actually was dealt.

Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction?
The player is not dependent on the deck according to the class diagram, which should be correct, but does not match the code.

Is the dependency between controller and view handled? How? Good? Bad?
The view has no more dependencies from the controller than before implementations and this is true vice versa. This is good as reducing dependencies helps high cohesion and low coupling.

Is the Strategy Pattern used correctly for the rule variant Soft17?
Yes. The Strategy Pattern requires a single interface for the strategy, which can then be utilised as soft17 or the standard rules as required. As there is still the basic hit strategy and a new class of this strategy "Soft17HitStrategy", it is correct.[1]

Is the Strategy Pattern used correctly for the variations of who wins the game?
Yes. As per above, there is a single interface with multiple variants, one for dealer winning on a draw and one for the player winning on a draw. This obeys the Strategy Pattern as they are now independently interchangeable from the client function.[1]

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?
Duplicate code is removed, however, as far as I could tell, the player class was not aware of the deck element. This means that by adding this function to the player class, another dependency is added.

Is the Observer Pattern correctly implemented?
A subscriber is set up in the controller if a card is dealt and this results in an alteration to the view. This abides by the observer pattern described here[2]. An added bonus is the nicely implemented "DisplayWait" function which lets the user know that work is being done and the program has not crashed.

Is the class diagram updated to reflect the changes?
The observer pattern appears as a function within classes "Game" and "PlayGame" passing to the view, as is done by the code.

Do you think the design/implementation has passed the grade 2 criteria?
Yes (though the dependency for the deck by the player could be avoided).

1. Gamma, Helm, Johnson & Vlissides (1994). Design Patterns (the Gang of Four book). Addison-Wesley. ISBN 0-201-63361-2
2. http://www.oodesign.com/observer-pattern.html