Workshop 3 peer review for Janty Azmat

Found problems

-

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

Yes. Since controller does not need to know detailed implementation of which string is entered rather than it needs to know which sequence to perform afterwards, dependency between controller and view is handled correctly. We would also suggest another solution which is to use enumeration, as we have limited choices: play, stand, hit and quit.

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

The strategy pattern inside the factory method is used as it is intended.
However, we would like to point out that usage of

```
int cardScores[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 10 ,10 ,10, 11};
cardScores[12] = 1;
```

could be replaced just by deducting 10 if there is an ace.

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

Yes. is-dealer-winner method of the dealer class is returning its value according to 'IWinGameStrategy' interface.

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?

Yes, new-game-strategy classes are not depending on deck. Repetition of get-card from deck and show card is replaced to hit method of the dealer.

Is the Observer Pattern correctly implemented?

The Player class is notifying the changes to INewCardDealtObserver class. Yet, subject of the observer pattern usually stores observers in a list, instead of a single-tone instance[1].

Is the class diagram updated to reflect the changes?
Yes.

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

Other comments
1. private final int g_maxScore = 21; of the IWinGameStrategy could have been avoided since dealer is having the same information. If the max score can be passed as an argument of the method ‘IsDealerWinner’ it will make programme more flexible if max score is changed in the future.

References

1. Microsoft Developer Network, Observer Design Pattern, 2016-11-01, https://msdn.microsoft.com/en-us/library/ee850490(v=vs.110).aspx