Peer Review by Daniel Svensson on Workshop 3 by Joakim Holmevi

I’m am using the e-book version of Larman’s *Applying Uml and Patterns* which, annoyingly enough, doesn’t have pagination corresponding to the printed book. Therefore I include both the e-book (pdf) page number and the section or figure where the reference can be found.

Test the runnable version of the application in a realistic way.

Dealer doesn’t stop on hard 17 after soft 17. I.e., if Dealer after having soft 17 (e.g. Ace and Six) gets a card with value 10, putting the Dealer at hard 17, the Dealer still takes another card.

Do the implementation and diagrams conform?

I can’t access the diagrams because they are only available through Visual Studio but not the free version which I have. No external copies, e.g. png or pdf, are available

Is the dependency between controller and view handled?

The hidden dependency has been removed. Instead an enumeration with gameplay actions have been added to the controller. The view returns one of the values to the controller depending on keypresses and the controller acts accordingly. However, by placing the enumeration in the controller a dependency from the view to the controller is added which results in a bidirectional dependency. If the enumeration is placed in the view, preferably the IView interface, the extra dependency could be avoided and low coupling supported [1, sect 17.12, p444].

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

Yes. The Soft17 rule is contained by a class interfaced by the IHitStrategy interface making it interchangeable in the application [1, sect 26.7, p613].

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

Yes. A new strategy interface called IWinsOnEqual is implemented by PlayerWinsOnEqual and DealerWinsOnEqual making them interchangeable [ibid]. The decision of which is used is made in the rules factory, returning an instance of IWinsOnEqual instead of the concrete class. Good.

Is the duplicate code removed from everywhere and put in a place that does not add any dependencies?

Yes the duplicate code has been removed. The dealer knows about the deck and accordingly gets a card and sends it to the Player DealCard method (either the dealer or the player). This supports information expert [1, sect 17.11, p439]. The new game interfaces have been updated to make use of this. Good solution.

Is the Observer Pattern correctly implemented?

Yes (the github comment says no but it's there?). The Player implements an observable interface, ISubject, and the controller implements the IObserver interface, both with necessary methods in accordance with the observer pattern [1, sect 26.10, p630]. The only reservation I have is naming the method in the IObserver interface UpdateCardInHand since this implies a very concrete, singular use of the method compared to something generic like Update(). However this is the only requirement for this particular workshop so it's not really a problem.

Is the class diagram updated to reflect the changes?

Can't access class diagrams.

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

As stated, I can't say anything about the diagrams; Other than that I think the design/implementation has passed.

References

1. Larman C. Applying UML and Patterns: An introduction to Object-Oriented Analysis and Design and Iterative Development. Third Edition. 2004. ISBN: 0131489062. E-book: https://aanimesh.files.wordpress.com/2013/09/applying-uml-and-patterns-3rd.pdf