Peer Review by Daniel Svensson on Workshop 3 by Sonny Kjellberg

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.

Note any problems/bugs.

Triple “GameOver”-message.
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.
The number of observers keeps adding up for every time you play a round (because the subscriptions is handled by the controller and not ‘emptied’ after the round ends).

Does 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 the view interface has an enumeration with gameplay values, one of which is sent to the controller depending on the key pressed in the view. Very good solution.

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]. However, a change in naming should be considered. IHitStrategy interface is implemented by the BasicHitStrategy and following that pattern of naming the soft17 class should be called something like SoftSeventeenHitStrategy. This is just an observation and not necessarily a part of naming conventions as realization and generalization is not the same thing.

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?

The duplicate code seems to be removed. It is however placed in the Player class creating a new dependency on Deck. The Dealer already knows about the Deck making it a better candidate in accordance with Information Expert and Low Coupling [1, sect 17.11, p439; sect 17.12, p444]. This would also translate better regarding a low representational gap since the player in the real world has no business getting cards, they are given to her by the dealer [1, sect 17.3, p416]. The affected interfaces have been updated.

Is the Observer Pattern correctly implemented?

Yes, with some reservations. The Player class implements a Subscriber interface (should probably be renamed Subject, Observable or Publisher since the Subscriber is “on the other end” of the pattern) and the controller implements an observer interface called BlackJackObserver in accordance with the Observer pattern [1, sect 26.10, p630].

The real problem is the fact that the controller assigns itself as an observer in the Play method which is looped by the Program, thus assigning itself over and over again each time a new round starts. This might not be a problem in an application on this scale but seeing as it would be a real problem on a more memory heavy application it is simply a bad practice. Also in subscribing to a Player the Player instance is actually returned from Game. I’m not an well versed in C# but since this returns the actual instance isn’t the encapsulation broken? E.g. the controller can now manipulate the Player and Dealer instance directly since “it’s a copy of a reference and it therefore points to the same instance that the original did, and so copying neither hides nor protects it from future manipulation” [2, p8].

The sleep function associated with the requirements for the observer implementation is located in the model rather than the view.

Is the class diagram updated to reflect the changes?

Can’t access diagrams.

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

Since I can’t access the diagrams I can’t answer that fully. The code seems to be up to it even if I think some things should be altered (card dealing responsibility and observer subscription).

References

1. Bain, S. L. Encapsulation as a First Principle of Object-Oriented Design. 2004. http://www.netobjectives.com/files/resources/Encapsulation_First_Principle_Object_Oriented_Design.pdf (Accessed 2015-10-25).
2. 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