Reviewer: mr222xu
Reviewee: aa223hs

Object Oriented Analysis and Design –

Workshop 3 Peer Review

Try to compile/use the source code provided. Can you get it up and running? Is anything problematic?

No problem compiling the source code. The program runs fine. The "pause" functionality when a card is dealt is implemented.

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

The diagram incorrectly shows that Player realizes Subject. However, Subject is an abstract class and not an interface. Thus should the relationship be of the generalization-specialization type and the line should be solid and not dotted [1]. The same applies to PlayGame and Observer.

The diagram also incorrectly shows that DealerWinsStrategy has a dependency [2] to IWinStrategy. Rather the diagram should show that DealerWinsStrategy realizes the IWinStrategy interface [3].

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

Yes, the dependency issue is handled and it has been handled by using the Strategy pattern [4]. I think it is a good approach. Myself I choose to have the getInput() method return an enumeration instead. But I believe you approach is better actually.

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

Yes, I have nothing to comment on regarding the implementation using the strategy pattern [4].

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

Yes, I have nothing to comment on regarding the implementation using the strategy pattern [4]. But I am missing the implementation where the player wins the game and added code is not being used anywhere.

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?

[1] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 260
[2] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 260
[3] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 263
[4] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 447
[5] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 463
[6] https://www.tutorialspoint.com/design_pattern/observer_pattern.htm, 2016-11-01
[7] https://en.wikipedia.org/wiki/Observer_pattern, 2016-11-01

Yes, the duplicated code is removed, the INewGameStrategy interface are updated and consolidated to the Dealer class, i.e. the class that knows about cards and the deck.

Is the Observer Pattern correctly implemented?

The result when running the program looks correct. However, the actual implementation looks a bit odd compared to what is described in e.g. the course literature [5] and also the web sites that are referenced in the code [6,7]. My interpretation, of what is described in the referenced ources, would not include a separate Subject class. But rather the Player class is the Subject class. Maybe a minor detail because the end result is more or less the same. You just have one extra layer.

Is the class diagram updated to reflect the changes?

Yes, but some minor details regarding relationship types. See previous remarks.

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

Yes, I do.

[1] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 260
[2] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 260
[3] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 263
[4] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 447
[5] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, page 463
[6] https://www.tutorialspoint.com/design_pattern/observer_pattern.htm, 2016-11-01
[7] https://en.wikipedia.org/wiki/Observer_pattern, 2016-11-01