

## Peer review 3 jh222jx

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

The code outputs debug information "Hello world!". Doesn't

2. Test the runnable version of the application in a realistic way. Note any problems/bugs.

The code is generally buggy when handling input. No validation of the input make the program crash. The game doesn't end when you go bust. The dealer can go bust when you go bust and then you win. Ex: my score 24, dealers score 25.

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

The class diagram is hard to read. The rules and strategy is not shown.

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

It's implemented with the use of matching enumerations. This is a good and recommended solution. The enumeration is placed in the controller which is good. (Missed it myself). I think it's important to validate input so the game doesn't crash.

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

Yes, the implementation is correct. Use of the rules doesn't change if the algorithm changes. However the logic is faulty. The dealer will hit on a hard 17 if it includes an ace (10, 6, ace).

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

Yes, the implementation is correct. Use of the rules doesn't change if the algorithm changes.

7. 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?

Dealer, AmericanNewGameStrategy and InternationalNewGameStrategy have not been update according to the "The code for getting a card from the deck, show the card and give it to a player is duplicated in a number of places. Make a refactoring to remove this duplication and that supports low coupling/high cohesion." demand in the instructions.

8. Is the Observer Pattern correctly implemented?

Yes the implementation is correct. "Define a "subscriber" or "listener" interface. Subscribers implement this interface. The publisher can dynamically register subscribers who are interested in an event and notify them when an event occurs." – Larman C. chapter 26.10. The control implements observer interface which is right. The code handling the observer is a bit of spaghetti code. It's hard to understand. Why put an observer in both player and dealer?

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

Partially. But the rules is not shown in the class diagram.

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

Yes I think the implementation pass the criteria. However I think you need to change the class diagram to display the rules.

## References

Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0-13-148906-2