Daniel Nilsson and Daniel Blixt

Peer review by Roy Nilsson, Mikael Eriksson and Oskar Klintrot

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

Everything complies at first try.

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

Everything works as expected except that you can't quit the game by pressing 'q'. In `PlayGame.Play()` the if-statement returns `true` if the user want's to quit, it should return `false`.

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

No, the design diagram lacks arrows for the associations and also lacks the dependency between `Dealer` and `Card`. The interface `IPlayerObserver` is not included but a non existing class named `PlayerObserver` is included. If the `PlayerObserver` is supposed to be the interface then it lacks the realization to `PlayGame`.

Code follows the sequence diagram perfectly.

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

Dependency is removed and implemented in the `Iview` instead with enum, It's good because it removes dependencies.

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

The Strategy pattern is used correctly regarding the soft 17 rule since the strategy is in an own class which shares a common interface (1, p. 447). When testing the rule also seems to work as it should. However consider if you have to check `nonAceScore`. If the dealer has score of 17 and has an Ace, the score of the rest of cards could not be other than 6.

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

The application follows the strategy pattern according to section 26.7 in the book (1, p.447).

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?

The duplicated code is removed without creating any dependencies. Good work.

Is the Observer Pattern correctly implemented?

The observer pattern is correctly implemented. However, it could be a good idea to check whether the subscriber already exists in the list of subscriber or not in order to avoid duplication of subscriptions.

Miscellaneous

`Thread.Sleep(1000);` is in the controller, low-level code should be handled in the view. Larman says that “the View is the UI Layer, and the Controllers are the workflow objects in the Application layer” (1, p.209).

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

If the design models gets updated to UML notation standard and up to date with the code then the implementation should pass grade 2.

References

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062