1DV607 WS3 Peer Review

Developers:

•Seldin Music, Henry Pap, Rasmus Gazelius Skedinger

•Contact Person username: smubc09

•Contact person email: smubc09@student.lnu.se

•Workshop result: https://github.com/onkelhoy/1dv607/tree/master/Workshop%203

Reviewers:

•Ulrica Skarin, ls222xp@student.lnu.se,

•Peter Andersson, pa222ku@student.lnu.se,

•Christian Trosell, ct222hv@student.lnu.se

Review:

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

We managed to get it running.

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

While there was no runnable version, a compiled version runs fine. The required functionality seems to be implemented; the user is able to 'stand' and there is a pause implemented. However, there is no pause when dealing the initial cards or when the user picks 'hit'. According to the requirements "The pause should be when any player (dealer or player) gets a card".

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

Yes, the implementation and diagram conform to a great extent. We do nevertheless believe that the arrow between PlayGame and IView could be replaced with a dependency arrow. We also miss some association arrows to the Subject class from Game, Dealer and PlayGame.

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

Authors have moved the logic of reading key-strokes (i.e. "p", "n" etc) from the Controller to the View so the dependency is removed. The letters are mapped to numbers that in turn are mapped to actions. Using numbers is great, it may increase flexibility (have different input values in different views), but may also be hard to read / understand (i.e. magic numbers).

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

Yes, it looks good. The implementations of the two hit strategies are behind the abstract class IHitStrategy which in turn is used by the RulesFactory. Changing of hit strategy is done in the RulesFactory and the Dealer is not concerned with what the specific, concrete class it recieves.

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

Yes, it is correctly used. Both PlayerAdvantageWinStrategy and DealerAdvantageStrategy implements the interface IWinStrategy. Dealer is not in any case affected by which winning algorithm is used. This since Dealer gets an instance of the RulesFactory and acts accordingly thereafter.

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?

In order to remove the duplicated code the authors have created a class; CardHandle.java which very well encapsulates the logic to handle cards. This class takes care of most of the duplicated code, foremost in AmericanNewGameStrategy and InternationalNewGameStrategy. We did, however, find duplicated code not taken care of - in class Dealer. Maybe this whole problem would have been more easily solved by putting the logic in a public method inside Dealer. This method could then have been called from both game strategies.

Is the Observer Pattern correctly implemented?

Authors have used an abstract class for the Observer and a concrete class for the Subject - instead of interfaces. Interfaces are what seems to be recommended for the observer pattern according to Larman [1, p.463-465]. However, we believe that the abstract classes in this case adapt very well to the needs of the application (and therefore is more than ok). Interfaces could however eventually give greater flexibility to the application. According to the description of the Observer pattern at Tutorialspoint [3] - the authors seems to be spot on with the implementation.

Is the class diagram updated to reflect the changes?

Yes, we believe that the updated diagram reflects the changes in the code and also uses correct UML notation according to the overview of Basic UML Class Diagram Notation [2].

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

The project is well structured and gives a very good impression. The authors have implemented all requirements, except for some missing duplicated code as mentioned above and the requirement to pause when player gets a card on “hit”. Also some questions about the arrows in the diagram (also noted above) should maybe be addressed a little further. To pass the grade 2 we think that missing requirements mentioned and discussed above should be implemented. Also worth to mention that none of us reviewers are closely familiar with the programming language used in this project, JAVA. With this said, we might have misunderstood or misinterpreted language-specific code-standards.

References:

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

2.Basic UML Class Diagram Notation [Internet]. Published [2016-09-15]. Available from: http://www.umich.edu/~eecs381/handouts/UMLNotationSummary.pdf

3.Design Patterns - Observer Pattern [Internet]. Published [2016-11-01]. Available from: https://www.tutorialspoint.com/design_pattern/observer_pattern.htm