To be reviewed: Erland Jönsson

By: Hatem Houssein, Austin Ponten and Alexander Spottka.

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

The source code can be compiled and is runnable.

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

The runnable version is well done; it asks for which strategy to use on Soft17, American or International, etc.

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

After a closer look on the differences between the diagram and the implementation we can say that the implementation and the class diagram are consistent. A slight difference is, that in the controller package holds a file named SetupGame.cs as well as a GameSetupView.cs in the view folder which don't appear in the class diagram. We assume that it was about to get updated in the class diagram but wasn't completed in time. Another minor misstake in the diagram is that the class BasicHitStrategy is used in twice. We think that The BasicHitStrategy class which implements the IwinnerStrategy should be StandardWinnerStrategy

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

The dependency between controller and view is handled in accordance with the MVC model. After the changes made the controller had no responsibilities from the view which is compliant with the MVC model.

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

Yes the existing interface IhitStrategy.cs was used and implemented in the Soft17HitStrategy class. The user has the option to choose if the dealer should use the Soft17Strategy or not. The decision will be handled during runtime.[1, pg. 613]

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

Yes. The same pattern which was used for the HitStrategy was also used for the decision of who should win the game [1, pg. 613].

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?

All of the duplicated was removed and implemented in the Dealer class.

Is the Observer Pattern correctly implemented?

The approach of this project was to put the IObserver in the model and implement it from the Dealer.cs and the PlayGame.cs classes. Everytime the dealer deals a new card the observers updates the Controller. This approach is what is expected from the Observer pattern and handled correctly.

Is the class diagram updated to reflect the changes?

As answered before there are some things not updated yet.

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

Yes, the main concepts of the Observer Pattern and the Strategy pattern were implemented correctly. There were a few minor flaws we found and mentioned before. These flaws should be corrected but in general we can say that all key requirements are met.

Reference List:

1. Craig Larman, Applying UML and Patterns 3rd Ed, 2004, ISBN: 0-13-148906-2