Peer review Workshop 3

For: Andreas Anemyr (aa223ig)
By: Ola Franzén (of222au)

In the course Objecktorienterad analys och design med UML

Starting up and get the source code running was easy and unproblematic. The game seems to function well and the pausing while cards are given to players is done well. I do notice though something that seems like a small (design?) bug, probably connected to the pausing when dealing cards. The first card the dealer and player gets are displayed as the same card in the beginning, but once the second card is given the correct cards are displayed. Looking through the code I notice that the SimpleView class, function DisplayYouGotANewCard() that calls the display of hands, gives the same hand to both dealer and player. Luckily this doesn't affect the game later on as the correct cards still are given to the players and displayed after the first ones.

The dependency between view and controller (the GetInput function mainly) has been handled okay, now at least the controller gets specific action back by enum. The view still returns the actual key input, but since the mapping of the keys to the enum is done in the view it can be considered okay. It could possibly be even better though to send the actual enum constant back, and have the enum as return value instead of int for safer and clearer handling in the controller. As of now the view can send any int back to controller so just looking at the view you don't really know what to send back to the controller.

The Soft17 rule variant has been solved with the strategy pattern in a good way. The same goes with the win rule variants; it has been solved well by the strategy pattern as described by Larman [1, p448]. The win rule strategy class takes a dealer and player object as parameters, it could possibly be done a bit less dependency needed if taking just the dealer and player score as parameters instead, making the eventual change of Player object not conflicting with the win rule strategy classes.

The duplicate code that was about getting cards from the deck and dealing them to players has been taken care of and put in a function in the Player class. This does however add more dependencies and an extra association to the Player class with the Game member variable. This breaks the Information Expert principle, described by Larman [1, p294] as "assign a responsibility to the information expert – the class that has the information". I think the information expert in this case when it comes to dealing cards should be the Dealer class – it has the deck and knows both the dealer (itself) and already uses Player as a dependency.

The construction of the dealing of cards described above also affects the observer pattern use for notifying the controller of cards that are dealt. Overall the observer pattern has been used correctly with the controller observing the Game model as described by Larman [1, p465]. However the solution with the Player class getting the Deck object as parameter for retrieving cards and calling the Game class to send notifications can be considered a bit bad since it adds more dependencies (and an association). This could rather easily be solved by using the same good observer pattern design used in Game also in the Player, or Dealer class. Then the Game could observe the Player (or Dealer) in the same way the controller observes the Game currently.

The updated class diagram looks good and covers the changes well, except maybe for a missing association from the Game to the IBlackJackObserver interface.

All in all I think all tasks have been solved, and the patterns to be used have been used in a good way. Maybe some fixing with the dependency problems could be needed though, and the grade 2 would be well passed.

References

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