Linnæus University
Course: Object Oriented Analysis and Design (1DV607, Autumn 2016)

Peer-Review of Workshop 3 (grade 2)

Reviewed work by: **Leif Karlsson**
Reviewer: **Mitja Tim Rijavec Bruneus**

Functionality

The program compiles and runs without a problem. All the required functionality seems to be present. The messages displayed can be a bit erratic (game instructions are sometimes but not always displayed when user action is required, dealer and player hands are sometimes displayed twice after choosing hit), a nice user interface was however not a requirement.

Agreement of the diagram with the implementation

The new hit strategy and rules for winning are included correctly. The new ICardDealtObserver interface has no dependency on Game and neither is Game a subclass of ICardDealtObserver as indicated on the diagram. The correct relationship should be Game uses ICardDealtObserver (dependency). The PlayGame class implements the interface ICardDealtObserver, this can be indicated by a Realization relationship (the controller package is however not a part of the new class diagram). The GameStrategyHelper class has dependencies on Card, Deck, Dealer and player which should be indicated on the diagram.

Implementation of Design Patterns

MVC

The model-view-controller pattern is implemented correctly. The model has no dependencies on the view and controller. The view is no dependencies on the controller and the controller only depends on an abstraction (IView interface). The hidden dependency between the view and controller has been removed.

The view responds directly to the user input and invokes game methods contained in the model. According to the Controller pattern (see Larman, p. 429-), it might be better if the controller handled the gameflow and updated the model. If all action is to be handled in the view, there is no need for the method IView::GetInput to be public.

Strategy Pattern

The soft17 hit rule and the new rules for winning are implemented correctly. The rule algorithms are interchangeable without the need to change the structure of the program (see e.g. Gamma, Helm, Johnson, & Vlissideses, p. 349-). BasicWinnerRule and DealerWinsOnEqualScoreWinnerRule are effectively the same, one of the classes could therefore be removed if it is not needed for demonstration purposes.

Observer Pattern

The Observer pattern is implemented correctly and the view is notified and updated automatically (see e.g. Gamma, Helm, Johnson, & Vlissideses, p. 326). The delay as implemented seems quite meaningless but it fulfils the requirements.

Code duplication

The code duplication in the implementations of the INewGameStrategy interface has been successfully removed but the dependency on Dealer, Player and Deck remains. The Dealer calls the NewGame method in a class implementing the INewGameStrategy and sends in the current Deck object, itself(Dealer) and the Player object as parameters. The same parameters are then sent to the DealCard method in the GameStrategyHelper superclass which then calls methods Deck::GetCard, Card::Show and Player::DealCard. Since it is the Dealer that has references to the Deck, the Player and itself as well as being a subclass of Player it would seem like a good idea if the Dealer would call these methods according to the Information Expert design principle (see Larman, p. 439).

Furthermore, the method GameStrategyHelper::DealCard has two signatures, taking a Player or a Dealer as an argument. This overloading is unnecessary as the class Dealer a subclass of the class Player and it would therefore suffice to only the method that takes a Player.

There is a lot of code duplication in SimpleView and SwedishView, maybe the duplicated code could be extracted to superclass. Alternatively, a resource bundle could be used to handle internationalization.

Overall evaluation

The solution fulfils almost all the requirements for the pass grade. The class diagram should, however, be updated to reflect all the changes.

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software, ISBN:0-201-63361-2
2. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062