**Instruction**

Look at the models, implementation and any accompanying documentation. Try to have an open mind and focus on trying to understand the materials as it is presented.

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

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

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

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

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

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

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?

Is the Observer Pattern correctly implemented?

Is the class diagram updated to reflect the changes?

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

**Problems/bugs.**

The code did compile and I did not find any bugs when testing the application.

**Dependencies**

There is still a dependency between the controller and the view. I would propose usage of an enum to exterminate the string dependency, and keeping the call to the *GetInput* method in the view, since user input is the responsibility of the view. In its current state, if the string for playing is changed in the view class *SwedishView* from “p” to “s” as in “spela”,problems will arise. Now you might be tempted to change the code in the controller class *PlayGame* so that the comparison is to the string “s” instead. However, now the application is very specific to the user interface in Swedish.

However, instead, if there is an comparison of strings in the view, which returns an enum type, the dependency is removed.

**Observer Pattern**

There is no specific class for the observer, instead a property of delegate System.EventHandler is user to call handleEvent, which resides in the View, causing a dependence between model class Dealer and the view. I therefore conclude that the observer pattern is not implemented correctly.

**Rule Variations**

I did not find any class specifically handling the soft17 variation. Instead, the code seems to be in the class *BasicHitStrategy* and always returns true if the dealer has an ace in its hand, meaning that if the dealer has an ace, it can always hit again. I therefore conclude that the soft17 variation is not correctly implemented. The variant for who wins seems correctly implemented.

**Refactoring/ Duplicate code**

The application still contains duplicate code in the classes which handle initiating a new game, in the namespace rules.

**Class Diagram**

I could not find an updated class diagram.

**Pass/Fail**

Several things need to be fixed in order to pass the grade 2 criteria.