# Peer review for fh222dt by kk222hk

**Running the code**
The code compiles without issues. The game runs and plays like it should with one exception. There is no pause and interface rerender when a player gets a card. To quote the requirements: ”When the event is handled the user interface should be redrawn to show the new hand (with the new card) and the game should be briefly paused, to make the game a bit more exciting”.

This is because you are not publishing the event to the observers in the Player class. You have a list of subscribed observers, but you do nothing with them.

**Diagram and code correlation**
Your dealer class has an association to IWinnerRule called m_winnerRule. If you look at the class diagram notation in Larman chapter 16.1, you see that an association should be marked in the class diagram with an arrow and a role name (m_winnerRule in this case).

I'd also consider marking the dependencies from IWinnerStrategy to Dealer and Player. According to Larman in chapter 16.11, ”receiving a parameter of the supplier type” creates a dependency that should be marked in the class diagram for clarity.

Otherwise, code and diagram are similar. It's good that you used the interface realization (Larman, chapter 16.12), so it's easy to see what is realizing the interfaces and what is dependent on them.

**Controller/view dependency**
It's good that you used an enumeration for dealing with the dependency. In addition to fixing the dependency, the controller becomes easier to read and understand.

You might also want to consider putting the keys in the views in constants. If you want to change what keys to use, you'd only have to change one place instead of two (string dependency).

**Strategy pattern**
This pattern is used correctly for both the hit strategy and the win strategy. In Larman chapter 26.7, he definition of the patterns solution as ”Define each algorithm/policy/strategy in a separate class, with a common interface”, which is what you have done with the IHitStrategy and the IWinnerStrategy.

**Duplication fix**
You fixed the duplication by refactoring into a new method in the Dealer called Deal(). I think that's a good solution, it doesn't add any dependencies and makes the code look better.

**Observer pattern**
As I mentioned in the beginning, you don't publish any changes to your observers, so you get no result from using the pattern. You have your publisher, and a list of subscribers. All that's left to do is call the observer method on all subscribers when a card is dealt to a player.

**Do you think the design/implementation has passed the grade 2 criteria?**
Make sure that the observer pattern work, and it definitely passes grade 2.

**References**

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062
2. https://coursepress.lnu.se/kurs/objektorienterad-analys-och-design-med-uml/workshops-2/workshop-3-design-using-patterns/