Workshop 3 peer review for Jovydas, Mantas, Jean-Pierre

Found problems

1. Model's player is printing out 'hello list world' on console.
2. Dependency between controller and view is not resolved.
3. Duplication of code 'get card' from deck, and show (true) and deal card has not been refactored in new game strategies, etc.
4. Pausing two seconds are performed inside the model, which is again violating the model-view separation. Take a look at the requirement again.
5. Soft17Strategy class is changing visibility of the card, which is totally unnecessary due to duplication of the operations in stand method of the dealer class.
6. Variable rules for determining the winner is not implemented in the programme; it only has a separate class named so, and is used nowhere neither inside dealer nor game. Besides, It is not understood the intention for placement of the win rule in 'Game' class which is not used.
7. Besides, PlayerWinsOnEqual class has a **STATIC** integer which should be avoided[2].
8. Stand method of the dealer is taking two cards from the deck, which removes two cards from the deck and the other card is not used.

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

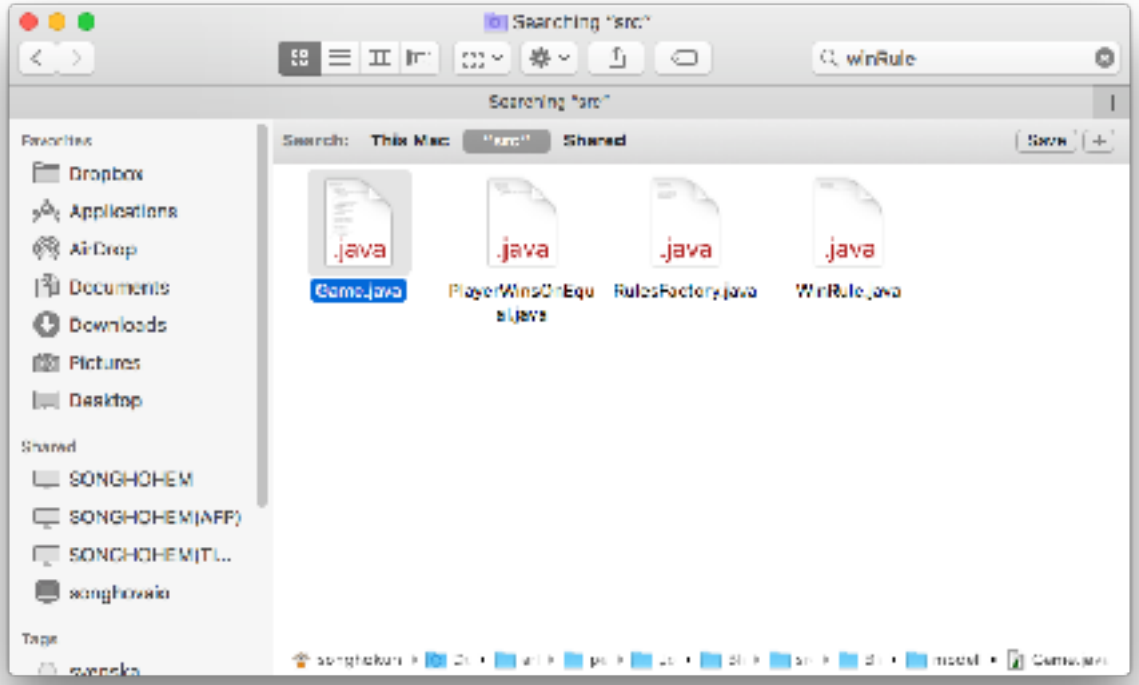
It is not handled, because controller is still depending on specific input form the view in a String form, which makes the controller dependent on specific type of view.

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

Yes, but we wonder if the current implementation of soft-17 is "using the same design pattern already present for hit." according the requirement. Tobias have suggested that we could extend 'BasicHitStrategy' class on the slack discussion.

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

No. It is not implemented in other model part at all. It only exists in these three classes which makes player wins on equal class and make it realise the interface, win rule class, despite it is used no where even inside the Game class.



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?

There are many code duplications. For example, Hit method inside the soft 17 strategy does not need to change the visibility of the card because it should be always done inside the stand method of the dealer. Besides, even stand method of the dealer has code duplications which is performing exactly same thing twice. Besides, refactoring of dealing card to player and dealer has not done; at least they seemed to try to avoid using 'get card' from deck and 'change visibility' by creating a method *'get visible card'*, however it is not called in new game strategy and so on. It could have been handled in a better way.

Is the Observer Pattern correctly implemented?

Yes for some extend[1]. However, providing the delay should be implemented in 'updateScreen' method, instead of having it in model.

Is the class diagram updated to reflect the changes?

Somewhat yes and no. So far, the class diagram applied changes of observer which showed realisation of the Observer from the controller. However, the dependency relations are missing from player class. Note that player has the array of the Observer's in its list. Besides their programme is using deck class both from the American new game and International new game, yet, dependency relation is not shown.

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

It has not implemented variable rules for who wins the game in model class, but just creating a class which does not have any associations with the rest of the model. Second, duplication of the getting card from the deck is not resolved. Third, implementation of the delay function is done not in proper way. For these reasons we believe this submission does not fulfil grade-2 criteria.

References

1. Microsoft Developer Network,Observer, 2016-11-01, https://msdn.microsoft.com/en-us/library/ff649896.aspx
2. T. Butler, "Tom Butler's programming blog," 12 Auguest 2012. [Online]. Available: https://r.je/static-methods-bad-practice.html. [Använd 11 October 2016].