605.201 MINI PROJECT #2

Program Design And Analysis

Guan Yue Wang

Johns Hopkins University

General Program Design

This program simulates a Black Jack game with one dealer and one player. The structure of the program consists of 3 classes: Card, Deck, and Player. To begin with, the Card class constructs a typical Poker card with card suit and number. Secondly, the Deck class uses the Card class to form a standard deck of 52 cards with different class methods to shuffle deck, deal cards, display all cards left in the deck, etc... Last but not the least, the Player class simulates the dealer and player in the game with all the relevant actions such as reset player hand, getting next card in the deck, and calculating the points on current hand.

The overall program is controlled by a while loop with termination upon one of the 3 conditions: player terminates the game, player runs out of money, or there's no more card in the deck. At each iteration, player and dealer each gets 2 cards. When nobody wins or loses, player gets to choose either he wants to hit more cards or stay as is. After that, dealer hits the cards till he reaches 17 or above. Afterwards, results are generated based on points value calculated based on each of their hands. At the end of the iteration, player would have the option to play another game if he still has money left.

One thing worth mentioning is the use of exception handler in the program. As we want to validate if there's still card left in the deck to get next card, try catch statement is utilized for each card dealt. Therefore, whenever there's no card in the deck, the program would be terminated and a customized 'Game Over' message would be displayed to tell user there's no more card in the deck for the array out of bound exception.

Alternative Approaches

One of the alternative approaches I've considered is to create separate classes for player and dealer. The benefit of having separate class is to help distinguish objects better and have different actions for each object. However, it is rejected because the only difference between player and dealer is the way the card is shown during the game. Therefore, instead of having a separate class,

I end up creating an unique method to print dealer's hand so the user has the option to hide the first card during the game and reveals all cards after the game. Consequently, I only need to use a different method to display dealer's hand instead of creating a dealer class in the design. However, another possible solution would be generating dealer class as a subclass of player class and then override the print hand method.

In addition, while I'm using exception handler to terminate the program whenever there's no more card in the deck and array out of bound exception occurs, an extra condition in the while loop is also considered to help end the program when running out of cards. However, the challenge is the while loop only checks the condition at the beginning of the iteration whereas the condition should be checked every time a card is dealt during the game. As a result, try catch statement at each card dealt for array out of bound exceptions becomes a better choice under this specific case.

Conclusion

By doing this project, I learnt the benefits and advantages of object oriented programming. By having objects set up in the classes, the phrase and syntax in the main game simulator becomes much easier to write and follow. To be more specific, encapsulation prevents the potential risks that deck gets mistakenly modified in the main program. Also, inheritance enables one to utilize the Card class for the construction of Deck class. Last but not the least, polymorphism gives the option to design dealer as a subclass of Player class. All in all, it is truly amazing how object-oriented programming is implemented for the design of this Black Jack game simulator.

UML Class Diagram