**Project overview:**

This project 1, we used java to create a blackjack game using basic java knowledge such as ArrayList, creating object in java and accessing associated data from the objects. The results of the project is to simulate the blackjack game with different types of rules, card deck numbers, and test out the overall possibility of winning for the player and the dealer.

**Solutions:**

1. I first created the Card class, whose function is merely to create a Card Object with index i as specified in the parameter. Then I created the Hand class, in which I first initialized an ArrayList that can only hold Card objects; then I created methods such as return the card value and returning the total value of card in hand. Finally, I crated the toString() method, where I formatted and printed out the indexes of all cards in the Hand ArrayList. The ToString method here is model for toString methods in other classes, which basically used the logarithm.
2. To create the Deck class, I first initiated an ArrayList that holds Card objects, then I created the build() method that runs a for loop and creates 52 Card objects and put them into the Deck ArrayList. The constructor for Deck class calls the build method. The shuffle method in Deck class also used for loop, where I created a new ArrayList, and randomly loop and remove every elements in the original Deck ArrayList and added the elements into the new list. In the end, I set the new ArrayList as the field list of Deck class.
3. In the Blackjack class, I first created three objects, which are respectively a Hand object for player, a Hand object for dealer, and a Deck object. I first wrote the constructor and the reset method, which clears the hand of player and the dealer and shuffle the cards in the deck.
   1. For the playerTurn function and the dealerTurn function, I used a while loop to keep the player/dealer draws card until the value of their cards meet the upper limit. These two functions will return Booleans, which will be used later to set the rule and play the game.
   2. In the game method, I first set a condition to reset and shuffle the deck if too few cards are left in the deck; then I used nested if statements to set the rule of the game according to the instruction on Wikipedia. The game method will return a integer value, which will be used in the simulation of the game.
   3. I tested the Blackjack class in the main function, where I created a Blackjack object and played the game for three times and printed out the state of cards that the player and the dealer have. It was at the end of the main function, I cleared the hand of the player and the dealer. Below is the result that got printed to the terminal when I played three games.
4. I stored the output from the three games to a txt file using the command line provided. mygames.txt is in my folder on the filer.
5. In Simulation class, I first created a Blackjack object and then a main function, where I used a for loop to execute the game for 1000 times. I also pre-set three integer field for the class, each of which represents the number of times the three result takes place (player wins, dealer wins, and push). At the end, I calculated and printed out the percentage that each result takes place over the 1000 times of game, and received the result below.

Extension

1. I first modified the rule section of the game (in the method game() in Blackjack class). I added a situation where if the player gets exactly 21 points while the dealer doesn’t, the player will win the round.
2. Then, I created the playerTurnInteractive() method in Blackjack class, in the replacement of the playerTurn() method, which simply forces the player to draw cards until the total values reach the limit. playerTurnInteractive() allows players to input from the terminal and decides whether they want to keep drawing cards. The method first creates a Scanner object to read from command line. If they answered “yes” to the question” Draw another Card”, they will be directed into the while loop to allow the player keep drawing cards until they answered “no” to the question, or the total value of cards in their hand exceed the limit of 21, where the while loop will break. Below is a snippet of the while loop and the input-reading algorithms.

For a user-friendly experience, I printed out the total value of cards in their hands before they make a decision of whether to keep drawing, and the value of card they draw, and the new total value of card indexes. To execute the playerTurnInteractive(), I crated another class Interactive to use the interactive method. And below is the result printed to the terminal when I execute the codes.

* 1. I also noted that in an occasion where the total value of player’s card exceed 21, the player will lose (bust) straightaway, and the dealer needn’t to deal cards because the dealer will win in this occasion. So, I created a Boolean print in Blackjack, which is default to true, but when the player’s hand exceeds 21, print will be set to false, which will directly lead to a player bust result without printing/dealing the cards of the dealer.

1. The next extension I did was creating a game with 6 deck of cards versus only one deck. The advantage of using more deck of cards in the real game is to prevent people from memorizing the value of the card presented and predicted the next card. I created a separate class for the 6 decks called BigDeck, which is also a child class of Deck class. I declared the inheritance relationship of BigDeck by putting “extends” in the class line, as shown below.

Thus, the class BigDeck will inherit all the methods from its parent class, Deck. Then, I override the constructor of BigDeck by also overriding the build method of Deck class, because it needs to contain 6 decks of cards. The build method of BigDeck calls its parent’s build method for 6 times, as shown below. Because all other methods of Deck are accessible to BigDeck, it became very easy for me to implement BigDeck into Blackjack by changing the deck field from Deck to BigDeck. Then I went back to the Simulator class to run the game for 1000 times, but using 6 decks of cards, I found out that the percentage of player win, dealer win, and push remain around the same value. But I found using 6 decks of cards produce a relatively more stable percentages of each result, while using only one deck of cards sometimes produce some extreme cases, after testing the program for several times.

**Questions**:

Are 1000 games sufficient to provide an accurate estimate of the win percentage for the player and dealer? What are those percentages?

* I think so. Because the ratio between the win percentages for the player and dealer remain relatively stable when simulating the game for 1000 times. I’ve also tried to simulate the game for 100 times, where the win percentages have much larger variations.
* The win percentage for the player is around 40%, while that for the dealer is around 50%; and the remaining 10% is the chances for push, when 1000 games are played.

**Summary**

In this project, I gained some basic knowledge of the algorithm and syntax of java, and some java build-in class such as ArrayList, Random and Scanner. Most importantly, this project has helped me a lot in transiting from python to java. I found it difficult at first because the algorithm of python is closer to verbal language form, but after respectively using java basic coding syntax in this project, I start to feel much more comfortable with the language.

**Credits**

Thanks to Professor Layton, Professor Maxwell, Roujia; TAs: Michael, Shailin, Mike; Wikipedia (Blackjack), Stock Overflow web page.