Omar Ahmadi

CS 111 Professor Tjang

Blackjack Project

When I first started the blackjack project, I did not know much about blackjack which made creating the project even more difficult. In the project I decided to use every class given in order to build up my way to blackjack.java. I started off with card.java and figured I would need to make an extra method to get the value of a card in blackjack (taking into consideration that jack, queen, king all equal 10). Afterwards, I made a few methods leading up to a toString method that I used to convert the integer values of faces and suits into a string. When working on Deck.java, I ran into my first major problem that was simpler than I thought. I spent a long time figuring out how to create a deck and shuffle it, without realizing I had it right since the beginning. Every time I tested the two methods I would get a weird return value (someone told me it was a spot in the memory) because I called the Card constructor in the opposite way. In Hand.java I used vectors instead of arrays because the size of any given hand is not known. The most important methods in this class were the addCard() and getCard() methods. Next, I made a class that extended Hand.java called BlackjackHand.java; this class was used to get the value of any given hand by calling a method. After complete these classes and methods, I went onto Blackjack.java and used bet, numPlayers, and money as static instance variables. After getting the user inputs, player names, and implementing the wagering system, I called the playblackjack() method within the main. The way I implemented multiplayer blackjack was with a max of 3 players, with each player going through their rounds individually versus the dealer. I tried having players play their rounds together, but I could not figure out a way for a player to retain their bet, name, and a way to find out whether the player won or not. I tried making separate arrays for each variable and even making a player constructor to hold these values, but I hit a brick wall along the way. In my code, the player start off with a bet then is dealt two cards along with the dealer. After outputting the hand, the user is able to decide whether to hit, stand, double down, or even split if both the cards in their hand were of equal face value. Based on user input and the cards added to their hands, I implemented a hint system which calculates basic probabilities and guides action from the user. I also tried to implement Card counting and found that a high/low method would have been optimal. I knew that every card on the table had to be counted with the cards 2, 3, 4, 5, 6 equal to 1, cards 7, 8, 9 equal to 0, cards Ace, 10, jack, queen, king equal to -1. I could not figure out how to make decisions based on the total value of the counted cards, so I was not able to implement the high/low card counting. When comparing the hands, if the user did not bust or obtain a blackjack, the hands of the players and the dealer are compared to see who wins the round. If the dealer and the player both have the same value hands (including blackjack), the dealer automatically wins. Along with the card counting, I did not do the GUI extra credit. For efficiency analysis, I counted the following operations: finding number of Players, player names, using wagering system, playing blackjack, getting user’s hand, and reading correct/incorrect input from user in terms of hitting standing and doubling down. I determined the worst case analysis (in big Oh terms) to be O(n^6).