Matt Rosenbaum

March 6, 2017

A Look Back at Blackjack

After looking back at my UML that I made before writing a single line of code, I see a lot of similarities and a lot of differences. For the most part, my final Card, Deck, Shoe and Hand classes all followed the original blueprint that I created in my UML. Specifically, with my Hand class, I was able to lay out a nice foundation because I knew a lot about blackjack before starting the project. I already knew that I would have to have methods checking for bust, blackjack, and five card charlie, so I didn’t have to change much in this class as I worked on it later. Although, my Player/Dealer classes were where things really went awry. When originally laying out the blueprints for these two classes, I envisioned them being pretty simple. I had a hit() method, a doubleDown() method, and a getInsurance() method in my Player class. My Dealer class only had a startGame() method, a deal() method, and a shuffleShoe() method. Yet, as I worked through the blackjack project and specifically these two classes, I realized how much more complex they would have to be. The Dealer class is most critical for game as a whole, as the dealer starts the game, ends the game, deals all of the cards, etcetera. As I realized this, I knew I had to make this class be much more detailed and specific. Instead of having just one deal() method, I had a deal() method, a dealFirst() method, a checkBlackjack() method, a dealPlayerNext() method, a dealDealerNext() method, and a displayHand() method. Having multiple methods really helped my code be more fluid and less brute. Each method narrowed in on a specific part of the game, whether it be dealing the first cards, dealing the player’s next cards, or dealing the dealer’s cards.

While there aren’t any specific bugs in my program that I can name, I did not use try/catch. So, if the player were to enter a double or a string when the scanner asks for a value, my game would crash. I make it blatantly obvious that the user should not enter a string or double when playing. I was able to make it so that if a player were to enter an incorrect value (like a bet that is greater than their bankroll), then they would be prompted to enter a new value.

I am very proud of my overall effort on the blackjack project. I spent a ton of time on the project, experienced a lot of errors, and learned so much from all of the trial-and-error. I really like how user friendly my game is and feel is flows pretty smoothly. Coming into the project, I was really familiar with blackjack and all of its rules, which really helped me write the proper code for the game. I was able to incorporate all scenarios that the game could present thanks to this prior knowledge.

In terms of going above and beyond, I spent a ton of time adding an offerInsurance() method. The offerInsurance() method was very challenging, but I was able to figure out. This method successfully prepares for any situation if the dealer’s first card was an ace and the player wanted to buy insurance. I needed a setBankroll() and setPot() method so that I could adjust the player’s bankroll value and bet value if the player wanted insurance. What was most difficult was that there are so many different possible outcomes of a hand, because you had to take into account if the player won insurance or won the hand, or neither. Therefore, there were many calculations and compilations that were needed to be made to help this method work properly.

It was so hard trying to account for all possibilities/outcomes that could occur in a blackjack hand. There are so many different ways that a hand could end, and if you forgot one, everything else would be thrown off. This happened to me many times. For example, in my code, I said that if the player’s hand has a greater value than the dealer’s hand, then the player wins. I was so confused when the player busted and it said the player won. It turns out that I forgot to write that the player must have a hand value of 21 or less for this to be true. I learned how to reprompt the user using a scanner and a while loop. If the user entered a value that was incorrect, I was able to tell the user their error and have them re-enter the value. I never utilized this skill throughout the year. If I had more time, I would have definitely loved to get into graphics. I really like playing the game of blackjack which is why I was able to get so invested into this project and genuinely enjoy it. Yet, I spent so much time with the coding and the scanner that I was never able to incorporate graphics.

I believe that I have earned an A on the blackjack project. I could talk about the obvious reasons, like how it is complete, thorough, and functions properly. Yet, I feel that I really deserve an A because of the passion I put into this project. My genuine interests for both coding and blackjack made for a perfect combination. There were days (like today, March 6),  in which I spent over five hours (It legitimately took that much time. It took a while to get offerInsurance() to work.) working on the code, yet the time seemed to fly by without me even noticing it. I spent so much time on this whole project and did not just try and fake my way through it. I didn’t try and get other people’s code and just have the right answers. I actually wanted to understand the code and wanted to work through my problems and difficulties on my own. I was able to work through a ton of problems without the help of others and it felt extremely rewarding that I was able to work my way through a challenge and build a game that I enjoy playing. This is what I was most proud of. I spent countless hours working on this project (as did everyone else), but was able to figure most things out on my own, which is a skill that I have developed more and more as the year has gone on. As a result, I was able to make a solid blackjack game, which is why I believe that I deserve to receive a great grade on this project.