Peyton Kelly

Professor Mohan

CMPSC 101

18 May 2021

Blackjack Fun

The reason for my project being blackjack meant a lot to me as I was implementing the code. For me, blackjack was important to my childhood because it was a game easy enough for a youngster like me at five years old to play. I remembered playing with my friends and neighbors on New Year's Eve and anywhere I went with my dad and his friends we would always play winning things like a bag of chips. Blackjack has influenced me to get into more card games and learn how to play to win. This game was the start of a "gambling" life because of this game there would be bets on everything like most kids. Making impossible shots in basketball to playing blackjack anything was being bet. Who hasn't bet their friends a million dollars on something that they insist they can do but they can not. Deciding to build a blackjack game is an important thing that has made me realize what is great about blackjack run on the computer. It gives people the opportunity to learn how blackjack is played. This program also makes an important to gamble responsibly when playing for actual money. This game using fake chips allows for someone to be able to gamble for fun and learn payouts and play as much as they want to learn what is best in playing blackjack. I feel that this project can give people a reliable way to spend their time responsibly while not losing any money and enjoying their time. This game can give to other people what it gave to me, joy and information of the game. This project is useful because it can be given to people as a way to pass time and have them learn about blackjack. This project has been an easy way to do something that is fun during this stressful time of finals.

Background

From this project, I have found similar blackjack games around but none that are quite like mine. Most people have been doing it by just showing the numbers and not having a deck. This makes the game and codes much more simple. Because of only showing the number there is no need to make a deck class because you can have repeating numbers. The way to get away with this is because if it seems you are getting the same numbers over and over again, the developer can just tell you there are multiple decks of cards which usually is true in casinos. This is because the number of players can range from one to eight plus one dealer at a table. To me, it is an easy cop-out, but the true reason this is mostly done because it can be done in quick lines of code. This can be done by generating random numbers, two for the player and two for the dealer. One thing that I would have liked to implement would have been the ability to have more than one player against the dealer, I have seen this done with the input from one user telling how many players are at the table and using for loops to iterate through each player x number of times. But for this to work there would have to be multiple decks and many layers on top of that within the betting system because it would have to be able to keep track of up to eight players' chips. This would be within different classes would have to use if statements and recall the chip amounts input and store the variables to differ between the different win methods such as push, bust, or straight win. For each, it would become problematic when coming to the hit and stand because it would end hands for certain players and would have to keep playing for the x number of players that could hit.

I believe that I will continue to work on this project to implement some improvements to the payouts. I would like to put in specials for good wins such as lucky ladys' meaning two queens of different suits. This would add some more fun to the game and make it more casino

like and that is what I am shooting for with this project. I already believe that I would be able to do this fairly easily. It would be coming in the payouts methods and add improvements to the payouts and make the if statements that correlate with the strings that would be produced in the hand class.

Data overview

Starting with my card variables, these are set up as strings for the print and display of the game. Once all the cards have been set up correctly as separate entities under the suits and rank variables. Then once those cards are set up it comes time to set a value for each card. This is done in a dictionary called values in my code. Setting up the Ace card as 11 is normal and is adjusted later in the code.

Beginning to set up my first class is the card class, this is an abstract class that is only trying to have set up the proper output of the display. Throughout my code is the idea of polymorphism because, in most of my classes, I am using def __init__ to set variables in my classes and methods. Having polymorphism within my code gives the ability for it to be adaptable to what I want to do with the game. I am able to change certain variables without having to change any of the actual implementations because they will all be changed in the above abstract methods.

Then it comes to all the cards rather than individual ones, my deck class sets up all the cards in a deck of 52 cards. This adds to the randomness of the shuffle and no games being the same. This involves the __str__ and __init__ methods to append all the cards in the deck using the for loop with ranks. Within this class is the deal function because the cards must come out of the deck to be dealt. Using the pop function within dealing the cards means that there are only 52 cards in the deck so no cards will repeat. Dealing cards one time will return the single card that is

pulled out of the deck by the computer. Having 52 cards means there are almost an infinite amount of ways the cards can be dealt. This gives what I would say makes the game so special the luck and randomness that one needs to get to twenty-one.

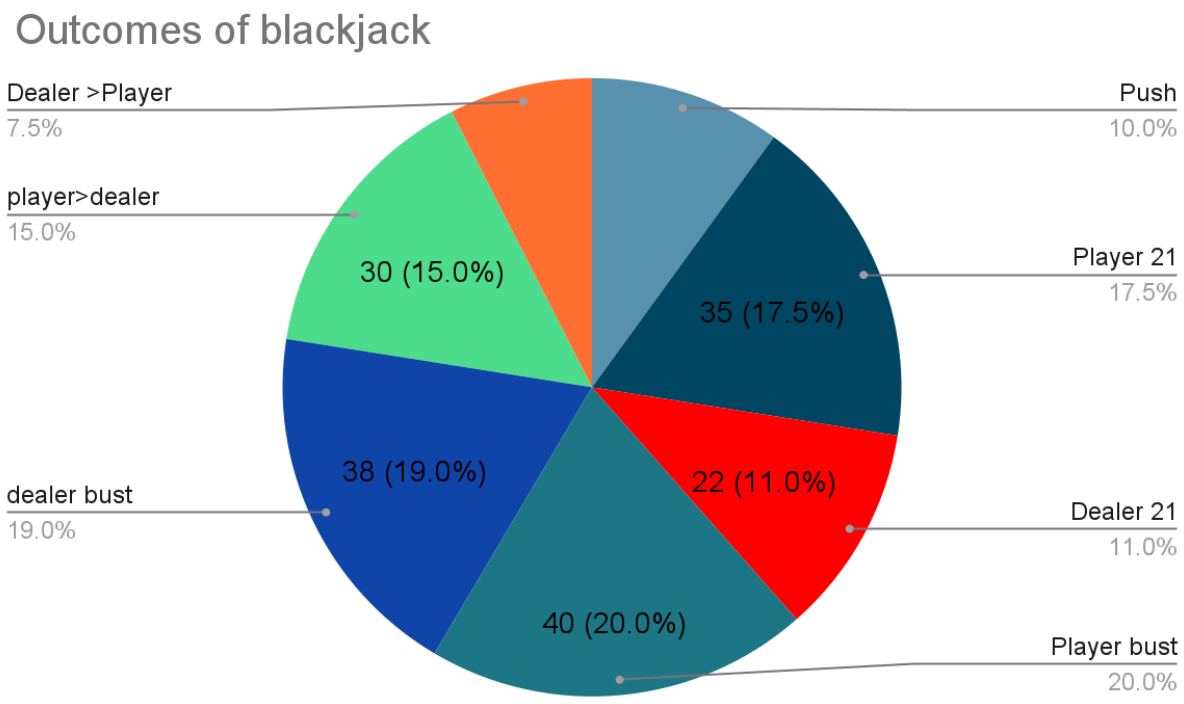
Next in classes is where things get serious if playing for actual monies. Creating the number of chips having a total of chips being two-thousand-five-hundred chips allows for a lot of fun being able to be had if the game was for real money. This class is also where the abstract methods are to pay out the winner's chips or take the chips if one loses the hand. These are set up simply to pay out 2:1, the most common payout in blackjack other than 6:5.

Once all these abstract classes are set up it comes down to all the implementations, including all the inputs that the user must make to play the game properly. While 'playing' the game is true, the system begins to interact with the player. Starting with the first input that is needed within the game will be the player's chip amount to be bet. This betting input is built for user error to repeat until the user gives a correct answer. Taking the bet makes sure that the wager is not more than the user starts with of 2500. With all the possible inputs to help out the user if a mistake is made, the program will help out and tell the user what they did wrong and how to correct it.

More abstract methods are set up for the crucial decision to hit or to stand. This adds to the hand class by adding an additional card to the hand. Within this method, it continues to call back to the adjustment for the ace. Then comes the implementation of playing the game. This prompts the user to make a decision after the first two cards are dealt. The decision makes the player gain another card or hold at the number they are at. This is also built in to fix all mistakes by the user.

After the final coding to make the game playable it comes to the presentation of the game. This allowed for me to get creative with the ways I wanted to display the game in the terminal. For the dealer's cards, one card needs to be hidden to have deception built into the game like normal. For those who are not familiar with the game, the dealer will deal their first card up and the second face down because it forces the player to make decisions with much uncertainty. This is why it is a game where the house wins many times. Casinos will have their pretty chandeliers. Once cards are displayed, the total of the player's and dealer's cards will be shown to the player. Then playing till the user stands, it will allow for the dealer to play and try to win if blackjack was not reached. Finally, the program will go through the scenarios of play to see the outcome and give or take away the chips bet.

Results



The results of play have come out by playing two hundred games within the program's final product. Coming to the conclusion that the game is relatively fair by and the computer does not just pull out blackjack every time. With the six different scenarios that can happen, it came out being that the player if playing odds correctly of when to hit and stand properly, the player should win more often in a larger sample size. Since the game of blackjack is not based on skill but luck, results may vary(Table 1 above).

Conclusion

One of the biggest takeaways from this project overall has been the amount of fun I have had building and playing the game. It brought back many memories of playing when I was younger. Also, this project was something I could be proud of. Once the coding was complete I began showing my friends what I built and had them play the game over and over and got positive feedback from it and some constructive feedback that helped me develop and make it better. Some of my biggest challenges were where to start with the classes. I started trying to build the playing method first but then realized that it was a bad idea to start with the ending, so I just went to the source. I found some of the blackjack games I had on my phone and tried to model it off of them while being original. I realized that along the way through playing them their inputs are fairly simple except for the fact that the game is not text, you can see every movement that is made in the game. One of the biggest rewards I felt from this project was not only the approval of peers that know nothing about computer science but when I first got a working output my eyes lit up with success. The first output was crude and not all the chips paid out to start but I was able to fix minor details to get a working game. Working on this project one of my goals was to gain an understanding of abstract methods and I feel that I have at least learned and gained the bare minimum of what they can do for me in the future. Having these

methods all throughout my project forced me to get to know what to do and how to implement them for my code.