Multiplayer Black Jack game

Here are the key features of my Black Jack game.

1. Multiplayer Support: The game will prompt you to specify the number of players.
2. Text-Based: The game is text-based. Players will interact with the terminal.
3. The game will follow standard black-jack rules.

Design and Implementation

1. Design:
   1. Game will prompt user to input the number of players
   2. The game will dynamically create instances of the ‘Player’ class
   3. The game will use 1 standard deck containing 52 cards
   4. The game will randomly select a card from the deck when dealing
   5. The game will deal out cards until each player has 2 including the dealer
   6. The game will show all the players cards along with the total
   7. Each player will be prompted to Hit or Stay until the total > 21 or they type Stay
   8. The dealer will continue to Hit until total is >= 17
   9. At the end of the game, Results will be printed out
2. Implementation:
   1. Number of players input will be stored in a variable num\_players

A screenshot of a computer

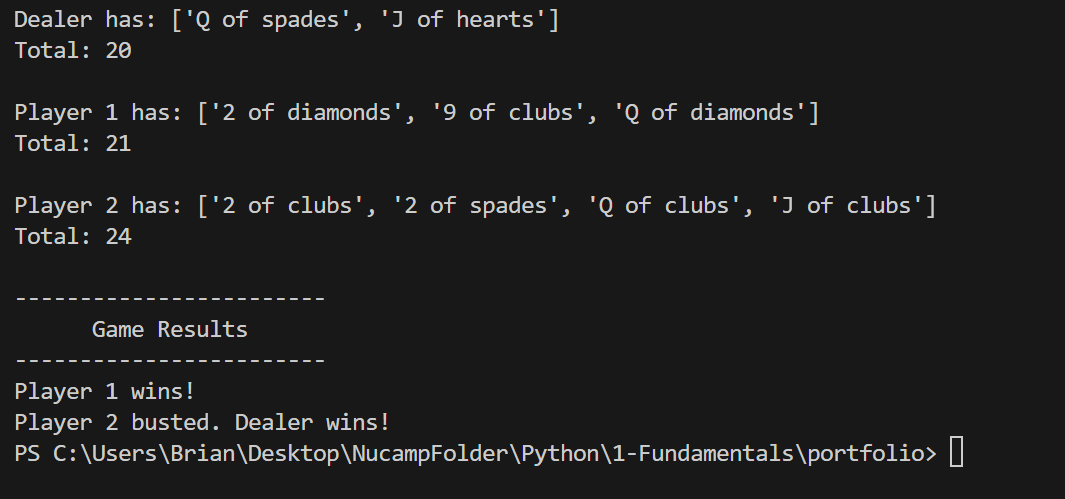
Description automatically generated

* 1. A list of ‘Player’ objects will be created using a for loop with num\_players as the stop value.
  2. The ‘Player’ class
     1. Attributes:
        1. Name: String
        2. Cards: Dictionary
        3. is\_bust: Boolean
        4. is\_dealer: Boolean
     2. Method:
        1. Sum\_of\_cards: returns the some of the values in Cards
  3. The Deck will be a dictionary data type representing a 52 card deck.
     1. The key will be the card ex: “A of spades”
     2. The value will be the value of the card ex: “A of spades” : 11
  4. The deal\_cards() function will take in an argument of ‘Player’ object
     1. get\_card() function will be called and return value will be assigned to player.card attribute
     2. get\_card()
        1. initialize an empty dictionary called card
        2. randomly select a card from card\_deck using random.choice()
        3. add card to card dictionary
        4. remove card from card\_deck
        5. return card
  5. The show\_cards() function
     1. Loop through players list
        1. Print player cards (Keys only)
        2. Print player card total
  6. After the initial 2 cards are dealt. Game will loop through player list and prompt ‘Hit’ or ‘Stay’ until the player types ‘Stay’ or the player busts(total > 21)

A screenshot of a computer program

Description automatically generated

* 1. Dealer will have the last action
     1. Dealer will automatically hit(get\_card) until total > 17
  2. The game will end
     1. Show\_winner() function will print the results of the game



1. Conclusion

This game was a great challenge and gave me the opportunity to apply what I’ve learned in the course. One of the features I like about the game is the ability to create multiple players. This added a little more complexity to the code but was a great experience to figure it out and actually apply to the game.

One of the challenges I faced when creating the game was trying to figure out the best way to implement game logic for handling “Soft Cards” which is a hand that contains an Ace that can be counted as either 1 or 11. Although I managed to get it to work, I feel there is a more efficient way to write the code to reduce the amount of conditional statements used.

In hindsight, I think creating a separate class for ‘Dealer’ would have been a better approach to the code. The function to deal cards would be a method in the Dealer class. Also the deck of cards could be a class on it’s own with methods to shuffle and assign cards.

A feature that could be added in the future would be the ability to actually place a bet and have multiple deck of cards.