Classes Challenge 37: Casino Black Jack App

Description:
You are responsible for writing a program that allows a user to play casino Black Jack. The user will put a set amount of money onto the table and make a minimum $20 bet each hand. Each hand, the user will be dealt two cards and be given the option to hit or stay. If the user hits 21 or goes over the round will end. The dealer will continue to hit until their hand has a minimum value of 17 as per casino guidelines. The user will be able to play as long as their total money is greater than or equal to the minimum bet of the table.

Step By Step Guide:
Defining your classes

- Define a class Card.
- Define a class Deck.
- Define a class Player.
- Define a class Dealer.
- Define a class Game.

Defining your class methods
The Card Class

- Define an __init__() method for the Card class which takes four parameters, the required self parameter, a rank, a value, and a suit.
    - Set the rank of the card to the given rank.
    - Set the value of the card to the given value.
    - Set the suit of the card to the given suit.

- Define a display_card() method for the Card class which takes one parameter, the required self parameter.
    - Print a statement describing the cards rank and suit such as “K of Hearts” .

The Deck Class

- Define an __init__() method for the Deck class which takes one parameter, the required self parameter.
    - Initialize an attribute called cards and set it equal to a blank list.

- Define a build_deck() method for the Deck class which takes one parameter, the required self parameter.
    - Create a list called suits that holds the four card suits.
    - Create a dictionary called ranks.
        - Each key should be the rank of a card: 2-10, J, Q, K, A
        - Each value should be the corresponding value of the card.
            - Numeric cards are their integer value.
            - J, Q, K are 10.
            - A is 11.

        - Use a for loop to loop through the suits.
            - Use a for loop to loop through the keys and values in the dictionary:
                - Create a card for the given rank, value and suit.
                - Append this card to the Deck's cards list.

- Define a shuffle_deck() method for the Deck class which takes one parameter, the required self parameter.
    - Shuffling the deck will consist of randomizing the elements of the cards list.
    - Type import random as the first line of code in your program.
    - Use the random library to shuffle the list of cards.

- Define a deal_card() method for the Deck class which takes one parameter, the required self parameter.
    - Pop a card off the list of cards associated with the Deck.
    - Return this card.

The Player Class

- Define an __init__() method for the Player class which takes one parameter, the required self parameter.
    - Initialize an attribute called hand and set it equal to a blank list.
    - Initialize an attribute called hand_value and set it equal to zero.
    - Initialize an attribute called playing_hand and set it equal to True.

- Define a draw_hand() method for the Player class which takes two parameters, the required self parameter, and a deck of cards.
    - Use a for loop to deal 2 cards to the player.
        - Call the deck's deal_card() method to get a card.
        - Append this card to the players hand.

- Define a display_hand() method for the Player class which takes one parameter, the required self parameter.
    - Print a message declaring the player's hand.
    - For each card in the players hand:
        - Call the card's display_card() method.

- Define a hit() method for the Player class which takes two parameters, the required self parameter and a deck of cards.
    - Call the deck's deal_card() method to get a card.
    - Append this card to the players hand.

- Define a get_hand_value() method for the Player class which takes one parameter, the required self parameter.
    - Set the hand_value attribute to 0.
    - Create a variable ace_in_hand and set it equal to False.
    - Use a for loop to loop through each card in the player's hand:

        - Update the current value of hand_value by adding the current cards value.
        - If the current cards rank is ‘A’:
            - Set ace_in_hand equal to True.
    - If the hand’s value greater than 21 and there is an ace in the hand:
        - Treat the Ace as if it were 1 instead of 11 by subtracting ten from the hands current value.
    - Print the total value of the hand.

- Define an update_hand() method for the Player class which takes two parameters, the required self parameter and a deck of cards.
    - If the value of the players current hand is less than 21:
        - Get user input for if they would like to hit.
        - If yes:
            - Call the players hit() method.
        - Else:
            - Set the player’s playing_hand attribute to False.
    - Else, the player either has blackjack or is over 21:
        - Set the player’s playing_hand attribute to False.

The Dealer Class

- Define an __init__() method for the Dealer class which takes one parameter, the required self parameter.
    - Initialize an attribute called hand and set it equal to a blank list.
    - Initialize an attribute called hand_value and set it equal to zero.
    - Initialize an attribute called playing_hand and set it equal to True.

- Define a draw_hand() method for the Dealer class which takes two parameters, the required self parameter, and a deck of cards.
    - Use a for loop to deal 2 cards to the dealer.
        - Call the deck’s deal_card() method to get a card.
        - Append this card to the dealer's hand.

- Define a display_hand() method for the Dealer class which takes one parameter, the required self parameter.
    - Prompt the user to press enter to reveal the dealer cards.
    - Use a for loop to loop through the dealer's hand.
        - Reveal an individual card by calling the cards display_card() method.
        - Pause the program for added suspense between cards.
            - To do this, you must import the time library.
                - Type import time as the second line of code in your program.
                - Pause the program for 1 second.
                    - To pause a program use the time library’s sleep() function.

- Define a hit() method for the Dealer class which takes two parameters, the required self parameter and a deck of cards.
    - The dealer must hit until they reach 17, then they must stop.
    - Get the dealers current hand value by calling the get_hand_value() method.
    - While the hand value is less than 17:
        - Get a card from the deck by calling the decks deal_card() method.
        - Append the card to the dealer's hand.
        - Get the current hands value.
    - Print how many cards are in the dealer's hand.

- Define a get_hand_value() method for the Dealer class which takes one parameter, the required self parameter.
    - Set the hand_value attribute to 0.
    - Create a variable ace_in_hand and set it equal to False.
    - Use a for loop to loop through each card in the player's hand:
        - Update the current value of hand_value by adding the current cards value.
        - If the current cards rank is 'A':
            - Set ace_in_hand equal to True.
    - If the hand's value greater than 21 and there is an ace in the hand:
        - Treat the Ace as if it were 1 instead of 11 by subtracting ten from the hands current value.

The Game Class

- Define an __init__() method for the Game class which takes two parameters, the required self parameter and an amount of money.
    - Initialize an attribute called money and set it equal to the given money as an integer.
    - Initialize an attribute called bet and set it equal to 20.
    - Initialize an attribute called winner and set it equal to a blank string.

- Define a set_bet() method for the Game class which takes one parameter, the required self parameter.
    - Create a variable betting and set it to True.
    - Run a while loop as long as betting is True.
        - Get user input for their bet.
        - If the bet is less than 20 set the game's bet attribute equal to 20.
        - If the bet is more than the game objects total money:
            - Inform the user that they cannot afford the bet.
        - Else:
            - Set the game's bet attribute to the users given bet.
            - Set betting to False.

- Define a scoring() method for the Game class which takes three parameters, the required self parameter, the player's hand value, and the dealer's hand value.
    - If the player's hand value is equal to 21:
        - Print that they got black jack.
        - Set the game's winner attribute to p.
    - Elif the dealer's hand value is equal to 21:
        - Print they got black jack.
        - Set the game's winner attribute to d.
    - Elif the player's hand value is greater than 21:
        - Print that the player went over 21.
        - Set the game's winner attribute to d.
    - Elif the dealer's hand is greater than 21.
        - Print the dealer went over 21.
        - Set the game's winner attribute to p.
    - Else:
        - If the player's hand is greater than the dealer's hand.
            - Print a summary.
            - Set the game's winner attribute to p.
        - Elif the dealer's hand is greater than the player's hand:
            - Print a summary.
            - Set the game's winner attribute to d.
        - Else:
            - It was a tie, print the summary.
            - Set the game's winner attribute to tie.

- Define a payout() method for the Game class which takes one parameter, the required self parameter.
    - If the game's winner attribute is equal to p:
        - Add the game objects bet to the game objects money.
    - Elif , the game's winner attribute is equal to d:
        - Subtract the game objects bet from the game objects money.

- Define a display_money() method for the Game class which takes one parameter, the required self parameter.
    - Print how much money the current game object holds.

- Define a display_money_and_bet() method for the Game class which takes one parameter, the required self parameter.
    - Print how much money the current game object holds and the current bet.

The main code

- Print a welcome message
- Print a message informing the user of the minimum bet of $20.
- Get user input for how much money they would like to place on the table.
- Create a Game() object called game.

- Create an active variable playing and set it equal to True.
- Use this variable to control a while loop:
    - Create a Deck() object called game_deck.
    - Build the deck by calling game_deck's build_deck() method.
    - Shuffle the deck by calling game_deck's shuffle_deck() method

    - Create a Player() object called player.
    - Create a Dealer() object called dealer.

    - Display how much money the game has by calling the game's display_money() method.
    - Get the bet for the game by calling the game's set_bet() method.

    - Draw the player's hand by calling the player's draw_hand() method.
    - Draw the dealer's hand by calling the dealer's draw_hand() method.

    - Display the games money and bet by calling the game's display_money_and_bet() method.
    - Print a statement that reveals the dealers first card.

    - While the player is playing their hand:
        - Display the player's hand by calling the player's display_hand() method.
        - Get the value of the player's hand by calling the player's get_hand_value() method.
        - Update the players hand (hit or stay) by calling the player's update_hand() method.

    - Have the dealer hit until they reach 17 by calling the dealer's hit() method.
    - Display the dealer's hand by calling the dealer's display_hand() method.

    - Score the round of black jack by calling the game's scoring() method.
    - Payout the round of black jack by calling the game's payout() method.

    - If the user has less than the minimum bet of $20:
        - Set playing equal to False
        - Print a message that the user ran out of money.

Example Output:

Welcome to the Blackjack App.
The minimum bet at this table is $20.

How much money are you willing to play with today: 100

Current Money: $100
What would you like to bet (minimum bet of 20): 20

Current Money: $100 Current Bet: $20
The dealer is showing a K of Clubs.

Player's Hand:
A of Clubs
2 of Diamonds
Total value: 13
Would you like to hit (y/n): y

Player's Hand:
A of Clubs
2 of Diamonds
J of Spades
Total value: 13
Would you like to hit (y/n): y

Player's Hand:
A of Clubs
2 of Diamonds
J of Spades
Q of Diamonds
Total value: 23

Dealer is set with a total of 2 cards.

Press enter to reveal the dealer's hand.
K of Clubs
10 of Spades
You went over 21....you loose!

Current Money: $80
What would you like to bet (minimum bet of 20): 40

Current Money: $80 Current Bet: $40
The dealer is showing a Q of Spades.

Player's Hand:
Q of Clubs
9 of Diamonds
Total value: 19
Would you like to hit (y/n): n

Dealer is set with a total of 3 cards.

Press enter to reveal the dealer's hand.
Q of Spades
4 of Diamonds
K of Diamonds
Dealer went over 21. You win!

Current Money: $120
What would you like to bet (minimum bet of 20): 110

Current Money: $120 Current Bet: $110
The dealer is showing a A of Clubs.

Player's Hand:
Q of Diamonds
7 of Clubs
Total value: 17
Would you like to hit (y/n): y

Player's Hand:
Q of Diamonds
7 of Clubs
J of Diamonds
Total value: 27

Dealer is set with a total of 2 cards.

Press enter to reveal the dealer's hand.
A of Clubs
Q of Hearts
The dealer got black jack, you loose!
Sorry, you ran out of money. Please play again.