```python
#Classes Challenge 37:  Casino Black Jack App
import random
import time

class Card():
    """Simulate a single card with rank, value, and suit."""

    def __init__(self, rank, value, suit):
        """Initialize card attributes"""
        self.rank = rank #2-10, J, Q, K, A
        self.value = value #1-11
        self.suit = suit


    def display_card(self):
        """show the rank and suit of an individual card."""
        print(self.rank + " of " + self.suit)


class Deck():
    """Simulate a deck of 52 individual playing cards."""

    def __init__(self):
        """Initialize deck attributes"""
        self.cards = [] #A list to hold all future cards in the deck


    def build_deck(self):
        """Build a deck consisting of 52 unique cards."""
        #Information for all potential cards in a deck
        suits = ['Hearts', 'Diamonds', 'Spades', 'Clubs']
        ranks = {'2':2, '3':3, '4':4, '5':5, '6':6, '7':7, '8':8,
                 '9':9, '10':10, 'J':10, 'Q':10, 'K':10, 'A':11,}

        #Build the deck, creating 52 individual cards and append them to the cards list.
        for suit in suits:
            for rank, value in ranks.items():
                card = Card(rank, value, suit)
                self.cards.append(card)


    def shuffle_deck(self):
        """Shuffle a deck of cards"""
        #Use random.shuffle() to shuffle deck
        random.shuffle(self.cards)


    def deal_card(self):
        """Remove a card from the deck to be dealt."""
        #Deal the last card in the shuffled deck
        card = self.cards.pop()
        return card


class Player():
    """A class for the user to play Black Jack."""

    def __init__(self):
        """Initialize the player."""
        self.hand = [] #A list to hold the players cards
        self.hand_value = 0 #Total value of the players current hand
        self.playing_hand = True #A bool to track if the player is playing the hand
```

```
63
64
65      def draw_hand(self, deck):
66          """Deal the players starting hand"""
67          #Player must start with 2 cards in hand
68          for i in range(2):
69              card = deck.deal_card()
70              self.hand.append(card)
71
72
73      def display_hand(self):
74          """show the players hand."""
75          print("\nPlayer's Hand: ")
76          for card in self.hand:
77              card.display_card()
78
79
80      def hit(self, deck):
81          """Give the player a new card."""
82          card = deck.deal_card()
83          self.hand.append(card)
84
85
86      def get_hand_value(self):
87          """Compute the value of the players hand."""
88          self.hand_value = 0
89
90          #Bool to track if you have an Ace
91          ace_in_hand = False
92
93          for card in self.hand:
94              self.hand_value += card.value
95              #Check for Ace
96              if card.rank == 'A':
97                  ace_in_hand = True
98
99          #The user went over 21, but they have an ace so treat ace as a 1.
100         if self.hand_value > 21 and ace_in_hand:
101             self.hand_value -= 10 #Ace is treated as 1 instead of 11 so subtract
    10 from hand_value.
102
103         print("Total value: " + str(self.hand_value))
104
105
106     def update_hand(self, deck):
107         """Update the players hand by allowing them to hit."""
108         #The player has the option to hit
109         if self.hand_value < 21:
110             choice = input("Would you like to hit (y/n): ").lower()
111             if choice == 'y':
112                 self.hit(deck)
113             #Player is happy with hand value, done playing hand
114             else:
115                 self.playing_hand = False
116         #Player is over 21, cannot hit again
117         else:
118             self.playing_hand = False
119
120
121 class Dealer():
122     """A class simulating the black jack dealer.  They must hit up to 17
123     and they must reveal their first card."""
124
125     def __init__(self):
```

```
126         """Initialize the dealer"""
127         self.hand = [] #A list to hold the dealers cards
128         self.hand_value = 0 #Total value of the dealers current hand
129         self.playing_hand = True #A bool to track if the dealer is playing the
    hand
130 
131 
132     def draw_hand(self, deck):
133         """Deal the dealers starting hand"""
134         #Dealer must start with 2 cards in hand
135         for i in range(2):
136             card = deck.deal_card()
137             self.hand.append(card)
138 
139 
140     def display_hand(self):
141         """Show the dealers hand one card at a time."""
142         input("\nPress enter to reveal the dealer's hand. ")
143 
144         #Show all cards in the dealer's hand
145         for card in self.hand:
146             card.display_card()
147             #Pause the program for 1 second to build suspense
148             time.sleep(1)
149 
150 
151     def hit(self, deck):
152         """The dealer must hit until they have reached 17, then they stop."""
153         self.get_hand_value()
154 
155         #As long as the hand_value is less than 17, dealer must hit.
156         while self.hand_value < 17:
157             card = deck.deal_card()
158             self.hand.append(card)
159             self.get_hand_value()
160 
161         print("\nDealer is set with a total of " + str(len(self.hand)) + "
    cards.")
162 
163 
164     def get_hand_value(self):
165         """Compute the value of the dealers hand."""
166         self.hand_value = 0
167 
168         #Bool to track if you have an Ace
169         ace_in_hand = False
170 
171         for card in self.hand:
172             self.hand_value += card.value
173             #Check for Ace
174             if card.rank == 'A':
175                 ace_in_hand = True
176 
177         #The dealer went over 21, but they have an ace so treat ace as a 1.
178         if self.hand_value > 21 and ace_in_hand:
179             self.hand_value -= 10 #Ace is treated as 1 instead of 11 so subtract
    10 from hand_value.
180 
181 
182 class Game():
183     """A class to hold bets and payouts"""
184 
185     def __init__(self, money):
186         """Initialize attributes"""
```

```
187             self.money = int(money) #Total money the user is playing with
188             self.bet = 20 #Minimum bet per hand is $20
189             self.winner = "" #No winner yet, no hand has been played
190 
191 
192         def set_bet(self):
193             """Get a users bet for a hand of black jack."""
194             betting = True
195             while betting:
196                 #Get a users bet
197                 bet = int(input("What would you like to bet (minimum bet of 20): "))
198                 #Bet is too small, set to min value
199                 if bet < 20:
200                     bet = 20
201 
202                 #Bet is too high, make them bet again
203                 if bet > self.money:
204                     print("Sorry, you can't afford that bet.")
205                 #Bet is acceptable, set bet and stop betting.
206                 else:
207                     self.bet = bet
208                     betting = False
209 
210 
211         def scoring(self, p_value, d_value):
212             """Score a round of black jack."""
213             #Someone got black jack 21!
214             if p_value == 21:
215                 print("You got BLACK JACK!!! You win!")
216                 self.winner = 'p'
217             elif d_value == 21:
218                 print("The dealer got black jack...You loose!")
219                 self.winner = 'd'
220 
221             #Someone went over 21.
222             elif p_value > 21:
223                 print("You went over 21...You loose!")
224                 self.winner = 'd'
225             elif d_value > 21:
226                 print("Dealer went over 21!  You win!")
227                 self.winner = 'p'
228 
229             #Other cases.
230             else:
231                 if p_value > d_value:
232                     print("Dealer gets " + str(d_value) + ". You win!")
233                     self.winner = 'p'
234                 elif d_value > p_value:
235                     print("Dealer gets " + str(d_value) + ". You loose.")
236                     self.winner = 'd'
237                 else:
238                     print("Dealer gets " + str(d_value) + ". It's a push...")
239                     self.winner = 'tie'
240 
241 
242         def payout(self):
243             """Update the money attribute based on who won a hand."""
244             #You won, you earn money
245             if self.winner == 'p':
246                 self.money += self.bet
247             #You lost, you loose money
248             elif self.winner == 'd':
249                 self.money -= self.bet
250 
```

```
251
252         def display_money(self):
253             """Display current money for the overall game"""
254             print("\nCurrent Money: $" + str(self.money))
255
256
257         def display_money_and_bet(self):
258             """Display the current money and bet for a game round."""
259             print("\nCurrent Money: $" + str(self.money) + "\t\tCurrent Bet: $" +
    str(self.bet))
260
261
262     #The main code
263     print("Welcome to the Casino Blackjack App")
264     print("The minimum bet at this table is $20.")
265
266     #Create a game object to keep track of bets, total cash, round winners, and
    payouts
267     money = int(input("\nHow much money are you willing to play with today: "))
268     game = Game(money)
269
270     #The main game loop
271     playing = True
272     while playing:
273         #Build a deck, populate it with cards, and shuffle.
274         game_deck = Deck()
275         game_deck.build_deck()
276         game_deck.shuffle_deck()
277
278         #Create a player and dealer
279         player = Player()
280         dealer = Dealer()
281
282         #Show how much money the player has and get the players bet
283         game.display_money()
284         game.set_bet()
285
286         #Draw the player and dealer hands
287         player.draw_hand(game_deck)
288         dealer.draw_hand(game_deck)
289
290         #Simulate a single round of black jack for the player
291         game.display_money_and_bet()
292         print("The dealer is showing a " + dealer.hand[0].rank + " of " +
    dealer.hand[0].suit + ".")
293
294         #While the player is playing, show hand, calc values, allow player to hit or
    stay
295         while player.playing_hand:
296             player.display_hand()
297             player.get_hand_value()
298             player.update_hand(game_deck)
299
300         #Simulate a single round of black jack for the dealer
301         dealer.hit(game_deck)
302         dealer.display_hand()
303
304         #Determine the winner and the payout
305         game.scoring(player.hand_value, dealer.hand_value)
306         game.payout()
307
308         #The user ran out of money, kick them out
309         if game.money < 20:
310             playing = False
```

```
311             print("Sorry, you ran out of money.  Please try again.")
```