-Project 1-
Blackjack

Made By:

Daanish Baig, BT18ECE067.

Sidharth Dinesh, BT18ECE066.

Advait Panda, BT18ECE026.

Objective:

The goal of this assignment is to code a game of blackjack for any number of players with the following allowed play options – Hit, Stay, Double Down and Split where Hit means to draw a card to add to your hand, Stay means to stop your play, Double Down means to take only 1 additional card at the start while doubling your bet and Split means to split your hand of a pair into two hands.

Methodology:

To start the game we need a shuffled deck. As we wish to make a game with n no of players we shall make a deck of 52 cards and randomly choose a card from said deck using random.choice(). We make a function called deck() which returns us a deck of all 52 cards. We also need n no of players hence we use an array of class Player where each object refers to a player and has the data of their hand(s),bet(s) and score(s). We assign each player a hand along with the dealer.

We run a for loop so as to go through each object of our players array. We use another loop to go through each hand of the player. To check whether a hand is still in play or not we use a while loop. This allows us to simply break out of the loop whenever required. The score and hand are displayed to the player and their bet is taken in. After this they are given the choice to Hit, Stay, Double Down and Split. If they choose to Hit we simply append a card to their hand and compute the new score. If they bust we make sure the hand is now not in play. If they choose to Stay we simply make it so that the hand is now not in play. If they choose to Double Down and have two cards only, we draw one card and append it to their hand, double their bet and compute their new score after which we make the hand not in play. If they choose to Split and have a hand of a pair only then we separate the two into two hands, drawing a card for each hand so that each hand now has two cards calculating scores for each and taking the bet for the new hand. Henceforth the total no of hands has increased with each being played independently. Here there is an exception. If the pair is of aces we do not allow the hands to be played further. Hence we make the new hands, calculate their scores and assign the bets but we do not let them remain in play. Also note that if either hand now draws a card with a value of 10 this is not considered as a blackjack but is rather a non-blackjack 21.

Now we need the dealer to complete his play. The dealer is made to hit as long as the score of his hand is less than 17. Finally we compare each player's score with the dealer's and assign their winnings accordingly. The player gets 2.5 times their bet if the get a blackjack or 2 times their bet if they win against the dealer or if the dealer busts. They lose their bet if they bust or lose against the

dealer. If they tie with the dealer they get back their bet. If the dealer gets a blackjack but the player has a non-blackjack 21 they lose their bet. After this we simply display their winnings.

Documentation:

- **Variables used –**

  - players – list of objects of class Player to hold all player related data.
  - n – int storing no of players
  - DECK – list obtained from deck()
  - play – Boolean to check whether a hand is in play or not
  - dealer_hand – list to store the hand of the dealer
  - scd – variable to store the score of the dealer's hand
  - ch – variable to store the play option chosen by the player
  - winnings – variable to store the total money won/lost by player

- **Functions used –**

  - def deck() – makes and returns a shuffled deck containing 52 cards
  - def score(hand) – takes a hand of cards and calculates the score of the hand. Dynamically changes the rank value of an ace depending on whether or not the hand would bust with the value 11. Returns a list containing the score and a string which says 'Blackjack' if score is 21 or the score in string format
  - def check_pairs(hand) – takes a 2 card hand and checks whether both the cards are equal
  - def not_aces(hand) - takes a 2 card hand and checks whether both the cards are equal to an ace
  - def double_down(hand,bet,scp,j) – facilitates the play option of doubling down by appending a card to the hand, doubling the bet and and computing the new score

- **Class definitions –**
  - Player: Methods – nil
  - Attributes – hand: List to store the various hands of a player. Initially has only one hand. Can have multiple hands due to concept of splitting. Each hand is a list as well
    - bet: Stores the bets associated with each hand
    - score: Stores the score associated with each hand

-Project 1-
Blackjack

Results:

```
Enter no of players 3
Score: 11 Hand: [(9, 'Hearts'), (2, 'Hearts')]

Enter your bet 34
Score: 11 Hand: [(9, 'Hearts'), (2, 'Hearts')]

Hit,Stay,Double Down or Split? H for Hit; S for Stay; D for Double Down; P for Split D
Score: 15 Hand: [(9, 'Hearts'), (2, 'Hearts'), (4, 'Cloves')]
Score: 18 Hand: [(10, 'Cloves'), (8, 'Diamonds')]

Enter your bet 45
Score: 18 Hand: [(10, 'Cloves'), (8, 'Diamonds')]

Hit,Stay,Double Down or Split? H for Hit; S for Stay; D for Double Down; P for Split H
Score: 28 Hand: [(10, 'Cloves'), (8, 'Diamonds'), ('J', 'Hearts')]
Score: 15 Hand: [(5, 'Diamonds'), ('J', 'Cloves')]

Enter your bet 55
Score: 15 Hand: [(5, 'Diamonds'), ('J', 'Cloves')]

Hit,Stay,Double Down or Split? H for Hit; S for Stay; D for Double Down; P for Split H
Score: 20 Hand: [(5, 'Diamonds'), ('J', 'Cloves'), (5, 'Hearts')]

Hit,Stay,Double Down or Split? H for Hit; S for Stay; D for Double Down; P for Split S
Dealer's turn
Score: [14, '14'] Hand: [(4, 'Cloves'), ('J', 'Diamonds')]
Score: 22 Hand: [(4, 'Cloves'), ('J', 'Diamonds'), (8, 'Diamonds')]
Total winnings of player 1 are 136
Total winnings of player 2 are -45
Total winnings of player 3 are 110
```

Discussion:

The usage of a object of class Player for each player greatly simplifies the code as we can simply make an array of n such objects and this would hold all the needed data. Without a class we would have had to make an array for bets, scores and hands which would have increased working of this could significantly.

Conclusion:

Hence we have successfully coded a game of blackjack that follows the ruleset defined along with handling the exceptions of the game. It allows for n no of players and also has a robust betting system in place.

-Project 1-
Blackjack

Program Listing:

```
import random


class Player():
    #used to make a player object containing hands and their bets and scores
    def __init__(self,l):
        self.hand=[l]
        self.bet=[]
        self.scp=[]


def deck():
    #function returns a randomly shuffled list(deck) of cards having rank and suit
    #rr-list of ranks;s-list of suits;d-deck of cards
    r=list(range(2,11))+['A','J','Q','K']
    s=['Hearts','Spades','Diamonds','Cloves']
    d=[(rank,suit) for suit in s for rank in r]
    random.shuffle(d)
    return d


def score(hand):
    #function that takes list of cards and calculates the score of the hand
    #s-score of hand;numa-number of aces;hand-list of cards;card-card from list hand
    s=0
    for card in hand:
        if card[0] in range(2,11):
            s+=card[0]
        elif card[0] in 'JQK':
            s+=10
        else:
```

```
            s+=11
    #ace can have a value of unity or 11 hence if score>21 value of ace is changed to unity
    numa=len([card for card in hand if card[0] is 'A'])
    while numa>0:
        if s>21:
            numa-=1
            s-=10
        else:
            break
    if s==21:
        return [s,'Blackjack']
    else:
        return [s,str(s)]


def check_pairs(hand):
    #functon to check wheter a given hand conatins only a pair
    if hand[0]==hand[1]:
        return True
    else:
        return False


def not_aces(hand):
    #function to check wheter a given hand contains a pair of aces
    if check_pairs[hand] and hand[0][0]=='A':
        return False
    else:
        return True


def double_down(hand,bet,scp,j):
    #function to execute the act od doubling down
```

```
    bet[j]*=2
    hand[j].append(random.choice(DECK))
    scp[j]=score(players[i].hand[j])
    print('Score:',scp[j][0],'Hand:',hand[j])

#Setting up the deck,no of players and their hands along with the dealer's
DECK=deck()
n=int(input('Enter no of players '))
players=[Player([random.choice(DECK),random.choice(DECK)]) for _ in range(n)]
dealer_hand=[random.choice(DECK),random.choice(DECK)]

#main game logic for player:displays score and hand and asks if they will hit, stay, double down or split
for i in range(n):
    play=True
    length=len(players[i].hand)
    j=0
    while j<length:
        players[i].scp.append(score(players[i].hand[j]))
        print('Score:',players[i].scp[j][0],'Hand:',players[i].hand[j])
        players[i].bet.append(int(input('Enter your bet ')))
        while play:
            print('Score:',players[i].scp[j][0],'Hand:',players[i].hand[j])
            if players[i].scp[j][0]>=21:
                break
            else:
                ch=input("Hit,Stay,Double Down or Split? H for Hit; S for Stay; D for Double Down; P for
Split ")
                if ch=='S':
                    play=False
                elif ch=='D':
                    if len(players[i].hand[j])==2:
```

```
            double_down(players[i].hand,players[i].bet,players[i].scp,j)
            play=False
        else:
            print('Cannot double down')


    elif ch=='P':
        if len(players[i].hand[j])==2:
            if check_pairs(players[i].hand[j]) and not_aces(players[i].hand[j]):
                players[i].hand.insert(j+1,[players[i].hand[j].pop[0],random.choice(DECK)])
                players[i].scp.insert(j+1,score(players[i].hand[j+1]))
                print('Score:',players[i].scp[j+1][0],'Hand:',players[i].hand[j+1])
                players[i].bet.insert(j+1,int(input('Enter your bet ')))
                players[i].hand[j].append(random.choice(DECK))
                players[i].scp[j]=score(players[i].hand[j])
                print('Score:',players[i].scp[j][0],'Hand:',players[i].hand[j])
                length+=1
            elif check_pairs(players[i].hand[j]):
                players[i].hand.insert(j+1,[players[i].hand[j].pop[0],random.choice(DECK)])
                players[i].scp.insert(j+1,score(players[i].hand[j+1]))
                print('Score:',players[i].scp[j+1][0],'Hand:',players[i].hand[j+1])
                if players[i].scp[j+1][1]=='Blackjack':
                    players[i].scp[j+1][1]=str(21)
                players[i].bet.insert(j+1,int(input('Enter your bet ')))
                players[i].hand[j].append(random.choice(DECK))
                players[i].scp[j]=score(players[i].hand[j])
                print('Score:',players[i].scp[j][0],'Hand:',players[i].hand[j])
                if players[i].scp[j+1][1]=='Blackjack':
                    players[i].scp[j+1][1]=str(21)
                length+=1
                play=False
```

```
                    j+=1
                else:
                    print('Cannot split hand')
            else:
                print('Cannot split hand')
        else:
            players[i].hand[j].append(random.choice(DECK))
            players[i].scp[j]=score(players[i].hand[j])
    j+=1

#main game logic for dealer:dealer will hit if score is below 17
scd=score(dealer_hand)
print('Dealer\'s turn')
print('Score:',scd,'Hand:',dealer_hand)
while scd[0]<17:
    dealer_hand.append(random.choice(DECK))
    scd=score(dealer_hand)
    print('Score:',scd[0],'Hand:',dealer_hand)

#final conditions for game conclusion
for i in range(n):
    winnings=0
    for j in range(0,len(players[i].hand)):
        if players[i].scp[j][1]=='Blackjack':
            winnings+=2.5*players[i].bet[j]
        elif players[i].scp[j][0]>21:
            winnings-=players[i].bet[j]
        elif scd[0]>21:
            winnings+=2*players[i].bet[j]
        elif scd[0]>players[i].scp[j][0]:
```

```
            winnings-=players[i].bet[j]
        elif scd[0]<players[i].scp[j][0]:
            winnings+=2*players[i].bet[j]
        elif players[i].scp[j][0]==scd[0]:
            winnings+=players[i].bet[j]
        else:
            winnings-=players[i].bet[j]
    print("Total winnings of player {} are {}".format(i+1,winnings))
```