OBJECT ORIENTED PROGRAMMING BANK ACCOUNT PROJECT

In [8]:
```python
class Account():
    def __init__(self,owner,balance=0):
        self.owner=owner
        self.balance=balance
    def deposite(self,dept_amt):
        self.balance=self.balance+dept_amt
        print(f"Added {dept_amt} to the balance")
    def withdrawal(self,wd_amt):
        if self.balance>=wd_amt:
            self.balance=self.balance-wd_amt
            print("Withdrawal accepted")
        else:
            print("Sorry non enough funds!")
    def __str__(self):
        return f"Owner: {self.owner} \nBalance: {self.balance}"
```

In [9]:
```python
a=Account("Sam",500)
```

In [10]:
```python
a.owner
```

Out[10]:
```
'Sam'
```

In [11]:
```python
a.balance
```

Out[11]:
```
500
```

In [12]:
```python
print(a)
```

```
Owner: Sam
Balance: 500
```

In [13]:
```python
a.deposite(100)
```

```
Added 100 to the balance
```

In [14]:
```python
print(a)
```

```
Owner: Sam
Balance: 600
```

In [15]:
```python
a.withdrawal(600)
```

```
Withdrawal accepted
```

In [16]:
```python
print(a)
```

```
Owner: Sam
Balance: 0
```

In [18]:
```python
print(a)
```

```
Owner: Sam
Balance: 0
```

TRY EXCEPT FINALLY

In [1]:
```python
def add(n1,n2):
    print(n1+n2)
```

In [2]:
```python
add(10,20)
```

```
30
```

In [3]:
```python
number1=10
```

In [4]:
```python
number2=input("Please provide a number:")
```

```
Please provide a number:20
```

In [ ]:

In [12]:
```python
try:
    # wnat to attempt this code
    # MAY HAVE AN ERROR
    result=10+10
except:
    print("Hey it looks like you aren't adding correctly")
else:
        print("Add went well")
        print(result)
```

```
Add went well
20
```

In [10]:
```python
result
```

Out[10]:
```
20
```

In [14]:
```python
try:
    f=open('testfile','r')
    f.write("Write a test line")
except TypeError:
    print("There was a type error!")
except OSError:
    print("Hey you have an OS Error")
finally:
    print("I always run")
```

```
Hey you have an OS Error
I always run
```

In [22]:
```python
def ask_for_int():
    while True:
        try:
            result=int(input("Please provide number:"))
        except:
            print("Whoops! That is not a number")
            continue
        else:
            print("Yes thank you")
            break
        finally:

            print("End of try/except/finally")
            print("I will always run at the end!")
```

In [25]:
```python
ask_for_int()
```

```
Please provide number:a
Whoops! That is not a number
End of try/except/finally
I will always run at the end!
Please provide number:g
Whoops! That is not a number
End of try/except/finally
I will always run at the end!
Please provide number:4
Yes thank you
End of try/except/finally
I will always run at the end!
```

ERROR EXCEPTION AND LOOP

In [27]:
```python
try:
    for i in ['a','b','c']:
        print(i**2)
except:
    print("General error! Watch out")
```

```
General error! Watch out
```

In [30]:
```python
try:
    x=5
    y=0
    z=x/y
except:
    print("Error!!")
finally:
    print("All done")
```

```
Error!!
All done
```

In [33]:
```python
def ask():
    while True:
        try:
            n=int(input("Enter a number"))
        except:
            print("Please try again! \n")
            continue
        else:
            break
    print("Your number square is:")
    print(n**2)
```

In [34]:
```python
ask()
```

```
Enter a numbera
Please try again!

Enter a number10
Your number square is:
100
```

THE BLACKJACK GAME - MILESTONE PROJECT

In [45]:
```python
import random
suits=('Hearts','Diamonds','Spades','Clubs')
ranks=('Two','Three','Four','Five','Six','Seven','Eight','Nine','Ten','Jack','Queen','King','Ace')
values={'Two':2, 'Three':3, 'Four':4, 'Five':5, 'Six':6, 'Seven':7, 'Eight':8, 'Nine':9, 'Ten':10, 'Jack':10,
        'Queen':10, 'King':10, 'Ace':11}

playing=True
```

In [46]:
```python
class Card():

    def __init__(self,suit,rank):
        self.suit = suit
        self.rank = rank

    def __str__(self):
        return self.rank+ " of "+self.suit
```

In [47]:
```python
class Deck:

    def __init__(self):
        self.deck = []
        for suit in suits:
            for rank in ranks:
                self.deck.append(Card(suit,rank))

    def __str__(self):
        deck_comp = ''
        for card in self.deck:
            deck_comp += '\n'+ card.__str__()
        return "The deck has: "+deck_comp

    def shuffle(self):
        random.shuffle(self.deck)

    def deal(self):
        single_card = self.deck.pop()
        return single_card
```

In [48]:
```python
test_deck = Deck()
test_deck.shuffle()
print(test_deck)
```

```
The deck has:
Four of Diamonds
Queen of Diamonds
Jack of Hearts
Six of Spades
Jack of Clubs
Four of Spades
King of Hearts
Seven of Hearts
Queen of Clubs
Seven of Diamonds
Ten of Hearts
Eight of Spades
King of Diamonds
King of Spades
Queen of Hearts
Two of Hearts
Four of Hearts
Ace of Spades
Seven of Spades
Two of Spades
Three of Hearts
Eight of Diamonds
Five of Hearts
Nine of Hearts
Eight of Hearts
Jack of Diamonds
Five of Spades
Three of Spades
Four of Clubs
Ten of Spades
Ace of Clubs
Five of Diamonds
Nine of Clubs
Nine of Spades
Two of Diamonds
Two of Clubs
Ace of Hearts
Ten of Clubs
King of Clubs
Three of Clubs
Ace of Diamonds
Nine of Diamonds
Six of Clubs
Eight of Clubs
Three of Diamonds
Six of Diamonds
Jack of Spades
Queen of Spades
Seven of Clubs
Ten of Diamonds
Five of Clubs
Six of Hearts
```

In [53]:
```python
class Hand:
    def __init__(self):
        self.cards = []
        self.value = 0
        self.aces = 0

    def add_card(self,card):
        #card passed in
        #from Deck.deal()-->single Card(suit,rank)
        self.cards.append(card)
        self.value += values[card.rank]

        if card.rank == 'Ace':
            self.aces += 1

    def adjust_for_ace(self):

        # IF TOTAL VALUE > 21 AND I STILL HAVE AN ACE
        # THEN CHANGE MY ACE TO BE A 1 INSTEAD OF AN 11
        while self.value > 21 and self.aces > 0:
            self.values -= 10
            self.aces -= 1
```

In [54]:
```python
zero = 0
one = 1
two = 2
```

In [55]:
```python
if 1:
    print('True')
```

```
True
```

In [57]:
```python
test_deck = Deck()
test_deck.shuffle()

#Player
test_player = Hand()
#Deal 1 card from the deck CARD(suit,rank)
pulled_card = test_deck.deal()
print(pulled_card)
test_player.add_card(pulled_card)
print(test_player.value)
```

```
Ten of Spades
10
```

In [58]:
```python
test_player.add_card(test_deck.deal())
```

In [59]:
```python
test_player.value
```

Out[59]:
```
20
```

In [60]:
```python
class Chips:

    def __init__(self,total=100):
        self.total = total # This can be set to default value or supplied by a user input
        self.bet = 0

    def win_bet(self):
        self.total += self.bet

    def lose_bet(self):
        self.total -= self.bet
```

In [61]:
```python
def take_bet(chips):

    while True:

        try:
            chips.bet = int(input("How many chips would you like to bet? "))
        except:
            print("Sorry please provide an integer")
        else:
            if chips.bet > chips.total:
                print("Sorry, you do not have enough chips! You have: {}".format(chips.total))
            else:
                break
```

In [62]:
```python
def hit(deck,hand):

    single_card = deck.deck()
    hand.add_card(single_card)
    hand.adjust_for_ace()
```

In [63]:
```python
def hit_or_stand(deck,hand):
    global playing # to control an upcoming while loop

    while True:
        x = input('Hit or Stand? Enter h or s')

        if x[0].lower() == 'h':
            hit(deck,hand)

        elif x[0].lower() == 's':
            print("Player Stands Dealer's Turn")
            playing = False

        else:
            print("sorry, I did not understand that, Please enter h or s only")
            continue

        break
```

In [64]:
```python
items = [1,2,3]
```

In [65]:
```python
for card in items:
    print(card)
```

```
1
2
3
```

In [66]:
```python
def show_some(player,dealer):

    # dealer.cards[0]

    # Show only ONE of the dealer's cards
    print("\n Dealer's Hand")
    print("First card hidden!")
    print(dealer.cards[1])

    # Show all (2 cards) of the player's hand/cards
    print("\n Player's hand:")
    for card in player.cards:
        print(card)

def show_all(player,dealer):

    # show all the dealers cards
    print("\n Dealer's hand:")
    for card in dealer.cards:
        print(card)
    print("\n Dealer's hand: ",*dealer.cards,sep='\n')
    #calculate and display value (j+k==20)
    print(f"Value of Dealer's hand is: {dealer.value}")

    # show all the players cards
    print("\n Player's hand:")
    for card in player.cards:
        print(card)
    print(f"Value of Player's hand is: {player.value}")
```

In [67]:
```python
def player_busts(player,dealer,chips):
    print("BUST PLAYER!")
    chips.lose_bet()

def player_wins(player,dealer,chips):
    print("PLAYER WINS!")
    chips.win_bet()

def dealer_busts(player,dealer,chips):
    print("PLAYER WINS! DEALER BUSTED")
    chips.win_bet()

def dealer_wins(player,dealer,chips):
    print("DEALER WINS!")
    chips.lose_bet()

def push(player,dealer):
    print("Dealer and Player tie! PUSH")
```

In [68]:
```python
while True:
    # print an opening statement

    print("Welcome To Blackjack")
    # create & shuffle the deck, deal two cards to each player
    deck = Deck()
    deck.shuffle()

    player_hand = Hand()
    player_hand.add_card(deck.deal())
    player_hand.add_card(deck.deal())

    dealer_hand = Hand()
    dealer_hand.add_card(deck.deal())
    dealer_hand.add_card(deck.deal())

    # Set up the player chip
    player_chips = Chips()

    # prompt the player for their bet
    take_bet(player_chips)

    # show cards (but keep one dealer card hidden)
    show_some(player_hand,dealer_hand)

    while playing: # recall this variable from our hit_or_stand function

        # prompt for player to hit or stand
        hit_or_stand(deck,player_hand)

        # show cards (but keep one dealer card hidden)
        show_some(player_hand,dealer_hand)

        # if player hand exceeds 21, run player_busts() and break out of loop
        if player_hand.value > 21:
            player_busts(player_hand,dealer_hand,player_chips)

            break


    # if player hasnt busted, play dealers hand until dealer reaches 17
    if player_hand.value <= 21:

        while dealer_hand.value < player_hand.value:
            hit(deck,dealer_hand)

        # show all cards
        show_all(player_hand,dealer_hand)

        # run different winning scenarios
        if dealer_hand.value > 21:
            dealer_busts(player_hand,dealer_hand,player_chips)
        elif dealer_hand.value > player_hand.value:
            dealer_wins(player_hand,dealer_hand,player_chips)
        elif dealer_hand.value < player_hand.value:
            player_wins(player_hand,dealer_hand,player_chips)
        else:
            push(player_hand,dealer_hand)


    # inform player of their chips total
    print("\n Player total chips are at: {}".format(player_chips.total))

    # ask to play again
    new_game = input("Would you like to play another hand? y/n")

    if new_game[0].lower() == 'y':
        playing = True
        continue
    else:
        print("Thank you for playing!")

        break
```

```
Welcome To Blackjack
How many chips would you like to bet? 300
Sorry, you do not have enough chips! You have: 100
How many chips would you like to bet? 50

 Dealer's Hand
First card hidden!
Seven of Diamonds

 Player's hand:
Eight of Spades
Three of Clubs
Hit or Stand? Enter h or ss
Player Stands Dealer's Turn

 Dealer's Hand
First card hidden!
Seven of Diamonds

 Player's hand:
Eight of Spades
Three of Clubs

 Dealer's hand:
Ace of Diamonds
Seven of Diamonds

 Dealer's hand:
Ace of Diamonds
Seven of Diamonds
Value of Dealer's hand is: 18

 Player's hand:
Eight of Spades
Three of Clubs
Value of Player's hand is: 11
DEALER WINS!

 Player total chips are at: 50
Would you like to play another hand? y/nn
Thank you for playing!
```