PseudoCode & Annotations

Pseudo:

Variables

customer_name = “ “

Balance= 0,0

Count=0.0

#Creating class BankAccount

class BankAccount

customer_name

balance

count

__init__(customer_name)

#initialize customer_name

#Generating account_num using generateAccountNumber() method

account_num = generateAccountNumber()

balance =0

#Calling display method

display()

#increment the count

count+1

/#This method creates random account Number and return the value

generateAccountNumber()

return 1+random.randint(0,1000)*1000+random.randint(0,1000)

addAmount(amount)

```
#add amount to balance
balance += amount
withdrawAmount(amount)
if(check for balance-amount should not be negative)
subtract amount from balance
else
print(Not sufficient balance)
display()
print(customer_name,account_num,balance)


#Create Empty Bank account Collection
BankAccountCollection = []

Variable to count of entered account
count = 1

#method to create new Bank Account
opennewAccount()
name = str(input("..."))
#Creating new BankAccount object with name
obj = BankAccounts(name)
Adding BankAccount to Collection
increment count Variable

#method to close account
closeAccount()
account_number = int(input(....))
```

```
try:
for i in range(count):
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
#remove back account Collection by calling remove method
BankAccountCollection.remove(BankAccountCollection[i])
decrease value of count
break the loop


except ValueError as ve:
print(Enter valid account number)
except IndexError as e
print(Account does not exist)

displayAccountDetails()
try:
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
break the loop

except ValueError as ve:
print(Enter valid account number)
except IndexError as e
```

```
print(Account does not exist)

depositeAmount()
try
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
amount = int(input(....))
check amount is greater than zero and add amount to account
break the loop


except ValueError as ve:
print(Enter valid account number)
except IndexError as e
print(Account does not exist)

withdrawAmount()
try
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
amount = int(input(....))
check amount is greater than zero and withdraw amount to account
```

break the loop

except ValueError as ve:

print(Enter valid account number)

except IndexError as e

print(Account does not exist)

Annotations:

Our code has 5 main steps with annotations as displayed on our Main.py file

1: To create a class BankAcccounts. Then define a function using __def opennewaccount__. This keyword is used in Python to initialize attributes of the class when an object of that class is created. This step is followed by initializing the balance as 0.

```
def opennewAccount():
    name=str(input("Please enter the name on the account: "))
    obj=BankAccounts(name)
    bankAccountsCollections.append(obj)
    global count
    count+=1
```

2: To create a function deposit such that the amount of money is taken by input using float and is then added to the balance. Then the amount deposited will be displayed using the print statement in the next line as shown in the code below:

```
#function to deposit  an amount by asking the user to input an account number in which to make the deposit
def depositAmount():

    global count
    try:
# then inside a for loop I traverse my bank account list and compare the bank # entered with the ones stored in the list
# if there is a match, I display that account info and ask for the amount to be deposited into it
        account_number = int(input("Please enter the account number for which you'd like to to make a deposit: "))
        for i in range(count):
            if bankAccountsCollections[i].getAccountNumber() == account_number:
                print("\nThe Account Details for the selected bank account are: ")
                bankAccountsCollections[i].display()
                amount = int(input("\nPlease enter the amount that you'd like to deposit: "))
                if(amount>0):
                    bankAccountsCollections[i].addAmount(amount)
                else:
                    print("\nThe amount to be deposited must be greater than 0!")
                break
```

3: To create another function withdraw in which is going to take float input for the amount to get withdrawn. Using an if condition here just to check if there's sufficient balance available to perform a withdrawal of any amount from the account. If the balance is not sufficient then our class will show "The amount to be withdrawn must be greater than 0!".

```
#function to withdraw an amount by asking the user to input an account number from which to withdraw
def withdrawAmount():

    global count
    try:
# then inside a for loop I traverse my bank account list and compare the bank # entered with the ones stored in the list
# if there is a match, I display that account info and ask for the amount to be withdrawn
        account_number = int(input("Please enter the account number for which you'd like to make a withdraw: "))
        for i in range(count):
            if bankAccountsCollections[i].getAccountNumber() == account_number:
                print("\nThe Account Details for the selected bank account are: ")
                bankAccountsCollections[i].display()
                amount = int(input("\nPlease enter the amount that you'd like to withdraw: "))
                if(amount>0):
                    bankAccountsCollections[i].withdrawAmount(amount)
                else:
                    print("\nThe amount to be withdrawn must be greater than 0!")
                break
```

4: To create our final function which is `displayAccountDetails` function. It will display the final balance of the account after withdrawal and deposit.

```
def displayAccountDetails():

    global count
    try:
        account_number = int(input("Please enter the account number for which you'd like to display its details: "))
        for i in range(count):
            if bankAccountsCollections[i].getAccountNumber() == account_number:
                print("\nThe Account Details for the selected bank account are: ")
                bankAccountsCollections[i].display()
                break
```

5: Lastly, to create an object of our class so that we can call all the functions with that class to execute our code.

```
#main function to be called from main containing the menu displaying the different choices for the user
def main():

    option=0
    while True:
        try:
            option = int(input("Welcome to the Banking System. Please select one of the following options: "
                               "\n1)Open new account "
                               "\n2)Close an account "
                               "\n3)Display all account details "
                               "\n4)Deposit an amount "
                               "\n5)Withdraw an amount "
                               "\n6)Exit "
                               ))

            if (option == 1):
                opennewAccount()
            elif (option == 2):
                closeAccount()
            elif (option == 3):
                displayAccountDetails()
            elif (option == 4):
                depositAmount()
            elif (option == 5):
                withdrawAmount()
            elif (option == 6):
                break
```