Name : Annapoornima S ¶

Roll No. : 225229101

Lab9. Object Oriented Bank in Python

Question1. Create a new class called Account.

In [12]:

```
class Account:
#"""" A class used to represent a type of account """"
    instance_count=0
    @classmethod
    def increment_instance_count(cls):
        print('creating new account')
        cls.instance_count+=1
    def __init__(self,account_number,account_holder,opening_balance,account_t
        Account.increment_instance_count()
        self.account_number=account_number
        self.account_holder=account_holder
        self.balance=opening_balance
        self.type=account_type
    def deposit(self,amount):
        self.balance+=amount
    def withdraw(self,amount):
        self.balance-=amount
    def get_balance(self):
        return self.balance
    def __str__(self):
        return 'Account[' + self.account_number +']-' + self.account_holder +

#Main:
acc1=Account('123','John',10.05,'Current')
acc2=Account('345','John',23.55,'Savings')
acc3=Account('567','Pheobe',12.45,'Investment')

print(acc1)
print(acc2)
print(acc3)


acc1.deposit(23.45)
acc1.withdraw(12.33)
print("Balance : ",acc1.get_balance())
```

```
creating new account
creating new account
creating new account
Account[123]-John,Current Account = 10.05
Account[345]-John,Savings Account = 23.55
Account[567]-Pheobe,Investment Account = 12.45
Balance :  21.17
```

Question2. Keep track of number of instances of Account We want to allow the Account class to keep track of the number of instances of the class that have been created. Print out a message each time a new instance of the Account class is created. Print out the number of accounts created at the end of the previous test program.

In [13]:

```
print('Number of Account instance created : ',Account.instance_count)
```

```
Number of Account instance created :  3
```

Question3. Create sub classes for Account class The aim of these exercises is to extend the Account class you have been developing from the last two chapters by providing DepositAccount, CurrentAccount and InvestmentAccount subclasses.Each of the classes should extend the Account class by: CurrentAccount adding an overdraft limit as well as redefining the withdraw method. DepositAccount by adding an interest rate. InvestmentAccount by adding an investment type attribute.

Question4. Add Properties to Account class Convert the balance into a read only property, then verify that the following program functions correctly:

In [14]:
```
class CurrentAccount(Account):
    def __init__(self , account_number , account_holder , opening_balance , c
        super().__init__(account_number , account_holder , opening_balance ,
        self.over_limit = -over_limit
    def withdraw (self,amt):
        if self.balance-amt < self.over_limit:
            print("WARNING : withdraw would exceed your limit" )
    def __str__(self):
        return super().__str__() + ' overdraft limit:' + str(self.over_limit)
```

In [15]:
```
class DepositAccount(Account):
    def __init__(self,account_number,account_holder,opening_balance,interest_
        super().__init__(account_number,account_holder,opening_balance,'depos
        self.interest_rate=interest_rate
    def __str__(self):
        return super().__str__()+' interest_rate:'+str(self.interest_rate)
```

In [20]:
```
class InvestmentAccount(Account):
    def __init__(self,account_number,account_holder,opening_balance,investmen
        super().__init__(account_number,account_holder,opening_balance,'inves
        self.investment_type=investment_type
    def __str__(self):
        return super().__str__()+' investment_type:'+str(self.investment_type
```

In [21]:
```
aco1=CurrentAccount('123','John',10.05,100.0)
print(aco1)
aco2=InvestmentAccount('567','phoebe',12.64,'high risk')
print(aco2)
aco3=DepositAccount('345','John',23.55,0.5)
print(aco3)
```

```
creating new account
Account[123]-John,current Account = 10.05 overdraft limit:-100.0
creating new account
Account[567]-phoebe,investment Account = 12.64 investment_type:high risk
creating new account
Account[345]-John,deposit Account = 23.55 interest_rate:0.5
```

In [22]:
```
acc1.deposit(23.45)
acc1.withdraw(12.33)
print('balance:',acc1.get_balance())
```

```
balance: 32.290000000000006
```

In [23]:
```
acc1.withdraw(300.00)
print('balance:',acc1.get_balance())
```

```
balance: -267.71
```

In [24]:
```
print('number of account instance created:',Account.instance_count)
```

```
number of account instance created: 6
```

Question5. Add Error Handling routines This exercise involves adding error handling support to theCurrentAccount class. In the CurrentAccount class it should not be possible to withdraw or deposit a negativeamount. Define an exception/error class called AmountError. The AmountError should take the accountinvolved and an error message as parameters. Next update the deposit() and withdraw() methods on theAccount and CurrentAccount class to raise an AmountError if the amount supplied is negative. You should beable to test this using:

Question6. Package all classes into a separate module

Question7. Convert Account as Abstract Class

Question8. Create History of Transactions using Lists

In [34]:
```
class BalanceError(Exception):
    """ The Balance will be invalid """
    def __init__(self, account):
        self.account = account
class AmountError(Exception):
    def __init__(self, account, msg):
        self.account = account
        self.message = msg
    def __str__(self):
        return 'AmountError (' + self.message + ') on ' + str(self.account)
```

In [38]:
```
class Account:
    """" A class used to represent a type of account """
    instance_count = 0
    @classmethod
    def increment_instance_count(cls):
        print('Creating new Account')
        cls.instance_count += 1
    def __init__(self, account_number, account_holder, opening_balance, accou
        Account.increment_instance_count()
        self.account_number = account_number
        self.account_holder = account_holder
        self._balance = opening_balance
        self.type = account_type
    def deposit(self, amount):
        if amount < 0:
            print('You cannot deposit negative amounts')
            raise AmountError(account = self, msg = 'Cannot deposit negative 
        else:
            self._balance += amount
    def withdraw(self, amount):
        if amount < 0:
            print('You cannot withdraw negative amounts')
            raise AmountError(self, 'Cannot withdraw negative amounts')
        else:
            self._balance -= amount
    @property
    def balance(self):
        """ Provides the current balance """
        return self._balance
    def __str__(self):
        return 'Account[' + self.account_number +'] - ' + \
               self.account_holder + ', ' + self.type + ' account = ' + str(
```

In [ ]:

In [39]:
```
class CurrentAccount(Account):
    def __init__(self, account_number, account_holder, opening_balance, overd
        super().__init__(account_number, account_holder, opening_balance, 'cu
        self.overdraft_limit = -overdraft_limit
    def withdraw(self, amount):
        if amount < 0:
            print('You cannot withdraw negative amounts')
            raise AmountError(self, 'Cannot withdraw negative amounts')
        elif self.balance - amount < self.overdraft_limit:
            print('Withdrawal would exceed your overdraft limit')
            raise BalanceError(self)
        else:
            self._balance -= amount
    def __str__(self):
        return super().__str__() + 'overdraft limit: ' + str(self.overdraft_l
```

In [40]:
```
class DepositAccount(Account):
    def __init__(self, account_number, account_holder, opening_balance, inter
        super().__init__(account_number, account_holder, opening_balance, 'de
        self.interest_rate = interest_rate
    def __str__(self):
        return super().__str__() + 'interest rate: ' + str(self.interest_rate
```

In [41]:
```
class InvestmentAccount(Account):
    def __init__(self, account_number, account_holder, opening_balance, inves
        super().__init__(account_number, account_holder, opening_balance, 'ir
        self.investment_type = investment_type
    def __str__(self):
        return super().__str__() + ', type: ' + self.type
```

In [42]:
```
acc1 = CurrentAccount('123', 'John', 10.05, 100.0)
acc2 = DepositAccount('345', 'John', 23.55, 0.5)
acc3 = InvestmentAccount('567', 'Phoebe', 12.45, 'high risk')
```

```
Creating new Account
Creating new Account
Creating new Account
```

In [43]:
```
print(acc1)
print(acc2)
print(acc3)
```

```
Account[123] - John, current account = 10.05overdraft limit: -100.0
Account[345] - John, deposit account = 23.55interest rate: 0.5
Account[567] - Phoebe, investment account = 12.45, type: investment
```

In [44]:
```
acc1.deposit(23.45)
acc1.withdraw(12.33)
print('balance:', acc1.balance)
print('Number of Account instances created:', Account.instance_count)
```

```
balance: 21.17
Number of Account instances created: 3
```

In [45]:
```
try:
    print('balance:', acc1.balance)
    acc1.withdraw(300.00)
    print('balance:', acc1.balance)
except BalanceError as e:
    print('Handling Exception')
    print(e)
```

```
balance: 21.17
Withdrawal would exceed your overdraft limit
Handling Exception
Account[123] - John, current account = 21.17overdraft limit: -100.0
```

In [46]:

```python
try:
    acc1.deposit(-1)
except AmountError as e:
    print(e)
```

```
You cannot deposit negative amounts
AmountError (Cannot deposit negative amounts) on Account[123] - John, curre
nt account = 21.17overdraft limit: -100.0
```

In [ ]:

```
```