Lab 14
OOP – BSDS – Spring 2022

Note: If Visio (drawing technical figures and diagrams) is available in the lab. Create UML class diagram of this case study, otherwise draw it neatly on paper and handover to TA. You have to code it as well in Python (save every class as separate module)

Consider the banking system. The bank has customers. The customers open account in the bank. The banking system manage customers and their accounts. Bank and customers perform transaction on the accounts, which are of different types. For example, customer deposit/ withdraw cash, cheque and other banking instruments. Bank debit customer account with fee, charges and credit their account with markup.

The bank account are of different types (discussed next) where, Account is an abstract type. Similarly, bank transaction are of different types where, Transaction is an abstract type. The details of classes are as under:

BankAccount: (abstract parent class, import date in this class)

Data Members: *account no, balance, account type, opening date*

Member Functions: *init, deposit, withdraw (abstract method), credit, debit, get balance, str (abstract method), get account type*

Class Members: *total accounts*

Function Details:

init: This function has two parameters balance (opening balance) and account type. Add one to total accounts and assign to account no. Assign account type and balance to relevant data member. Create object of *datatime* (for this *import datetime*, and create object by calling *now* function from *datetime* class) and assign to data member opening data.

deposit: This function has one parameters amount. Simply, **add** amount to balance

withdraw: This function has one parameters amount. Here simply, **subtract** amount from balance

credit: This function is same as deposit; the only difference is that deposit is called by customer logically and credit is called by banking system to credit interest etc. in the account

debit: This function is similar to credit; the only difference is amount is **subtracted** from balance

get balance: Return balance

get account type: Return account type

str: Return string concatenating account no, account type and balance

class level function get total accounts: return total accounts

CurrentAccount: (child class of *BankAccount* class)

Data Members: No data member required

Member Functions: *init, debit_bank_charges*

Function Details:

init: This function has one parameter balance (opening balance). Call init function of super with balance and '*CURRENT*' (account type)

debit_bank_charges: This function has no parameter, call debit function of super class with value 100

withdraw: Simply call to withdraw of super class

str: simply, return a call to str of super

SavingAccount: (child class of *BankAccount* class)

Data Members: No data member required

Member Functions: *init, credit_daily_commision*

Function Details:

init: This function has one parameter balance (opening balance). Call init function of super with balance and '*SAVING*' (account type)

credit_daily_commision: This function has no parameter, just call credit function of super class with amount = balance x 0.008%

withdraw: Simply call to withdraw of super class

str: simply, return a call to str of super

FixedDeposit: (child class of *BankAccount* class)

Data Members: No data member required

Member Functions: *init, credit_commission*

Function Details:

init: This function has one parameter balance (opening balance). Call init function of super with balance and 'FIXED' (account type)

credit_commision: This function has no parameter, just call credit function of super class with amount = balance x 0.12%

withdraw: print message **`"This is a fixed deposit account, withdrawal not allowed"`**

str: simply, return a call to str of super

SuperSavingAccount: (child class of SavingAccount)

Member Functions: *init, print_account, credit daily commission*

Class Members: Define class level constant member *MINIMUM_BALANCE* initialized with value **50000**

Function Details:

init: This function has one data member balance. Call init function of super with balance and balance and 'S_SAVING' (account type).

withdraw: This account requires minimum balance (**50000** already discussed). Therefore, check if balance - amount is more than equal minimum balance, deducted amount from balance and print withdraw message, otherwise print relevant message.

credit daily commission: Override super class function. Use interest rate 0.01

str: Call str of super and concatenate current class data members

OverdraftAccount: (child class of CurrentAccount)

Data Members: No data member required

Member Functions: *init, debit bank charges, withdraw*

Class Members: Define class level constant member *MINIMUM_BALANCE* initialized with value -**50000** means, customer can overdraft of Rs. 50000/- with extra bank charges.

Function Details:

init: This function has one data member balance. Call init function of super with balance and 'OD' (account type).

withdraw: This account requires minimum balance (-**50000** already discussed). Therefore, check if balance - amount is more than equal minimum balance, deducted amount from balance and print withdraw message, otherwise print relevant message.

debit bank charges: Override debit bank charges function of super class and charge Rs. 500 as bank charges instead of Rs. 100.

str: simply, return a call to str of super

Transaction: (abstract parent class, import date in this class)

Data Members: transaction no, transaction *date, amount, debit/ credit, transaction type*

Member Functions: *init, debit (abstract method), credit (abstract method), str (abstract method)*

Class Members: *total transactions*

Function Details:

init: This function has two parameters amount, and transaction type. Create datetime object and assign to the data member transaction date. Assign amount to data member amount. Assign debit/ credit to data member for debit/ credit. Add one to total transactions and assign to transaction no

str: Return transaction no + transaction date + amount + transaction type separated by \t

Cash Deposit: (child class of Transaction)

Data Members: No extra member required

Member Functions: *init, str*

Function Details:

init: This function has only one parameter amount. Call super init with amount and 'Cash Deposit' (transaction type). Call deposit function of account deduct amount

str: Simply call str of super

Cash Withdrawal: (child class of Transaction)

Data Members: cheque no

Member Functions: *init, str*

Function Details:

init: This function has two parameters amount, and cheque no. Call super init with amount and 'Cash Withdrawal' (transaction type). Assign cheque no to data member. . Call withdraw function of account deduct amount

str: Call str of super and concatenate cheque no with tab space

Cheque Deposit: (child class of Transaction)

Data Members: cheque no, bank

Member Functions: *init, str*

Function Details:

init: This function has three parameters amount, and cheque no and bank. Call super init with amount and 'Cheque Deposit' (transaction type). Assign cheque no and bank name to data member. Call credit function of account to deduct amount

str: Call str of super and concatenate cheque no and bank with tab space

Cheque Withdrawal: (child class of Transaction)

Data Members: cheque no, title, bank

Member Functions: *init, str*

Function Details:

init: The cheque is given to someone for payment instead of cash and the person has presented the cheque in his/ her bank, where it came back to customer's bank for clearance. This function has four parameters amount, and cheque no, title and bank. Call super init with amount and 'Cheque Withdrawal' (transaction type). Assign cheque no, title and bank name to data member. Call debit function of account to deduct amount

str: Call str of super and concatenate cheque no, title and bank with tab space

Write main program to demonstrate all the functionality.