**HEXAWARE CODING CHALLENGE**

*CC-5 LOAN MANAGEMENT SYSTEM*

*BY AYUSH PADHY*

## CLASSES

1. Customer Class:

```python
class Customer:
    def __init__(self, customerID, cName, emailAddress, phoneNumber, Address, creditScore):
        self.customerID = customerID
        self.cName = cName
        self.email = emailAddress
        self.phone = phoneNumber
        self.address = Address
        self.creditScore = creditScore
```

```python
1 usage
@property
def CustomerID(self):
    return self.customerID

1 usage
@property
def CName(self):
    return self.cName

1 usage
@property
def Email(self):
    return self.email

1 usage
@property
def PhoneNumber(self):
    return self.phone

1 usage
@property
def Address(self):
    return self.address

2 usages
@property
def creditScore(self):
    return self.creditScore

@CustomerID.setter
def CustomerID(self, customerID):
    self.customerID = customerID

@CName.setter
def CName(self, cName):
    self.cName = cName

@Email.setter
def Email(self, Email):
    self.email = Email

@PhoneNumber.setter
def PhoneNumber(self, PhoneNumber):
    self.phone = PhoneNumber
```

```python
@Address.setter
def Address(self, Address):
    self.address = Address

2 usages
@creditScore.setter
def creditScore(self, creditScore):
    self.creditScore = creditScore
```

2. Loan Class

```
6 usages
class Loan:
    def __init__(self, loanID, customerID, principalAmount, interestRate, loanTerm, loanType, loanStatus):
        self.loanID = loanID
        self.customerID = customerID
        self.principalAmount = principalAmount
        self.interestRate = interestRate
        self.loanTerm = loanTerm
        self.loanType = loanType
        self.loanStatus = loanStatus
```

```
3 usages
@property
def loanID(self):
    return self.loanID

3 usages
@property
def principalAmount(self):
    return self.principalAmount

3 usages
@property
def interestRate(self):
    return self.interestRate

3 usages
@property
def loanTerm(self):
    return self.loanTerm

3 usages
@property
def loanType(self):
    return self.loanType

3 usages
@property
def loanStatus(self):
    return self.loanStatus

2 usages
@loanID.setter
def loanID(self, loanID):
    self.loanID = loanID

2 usages
@principalAmount.setter
def principalAmount(self, principalAmount):
    self.principalAmount = principalAmount

2 usages
@interestRate.setter
def interestRate(self, interestRate):
    self.interestRate = interestRate
```

```
2 usages
@loanTerm.setter
def loanTerm(self, loanTerm):
    self.loanTerm = loanTerm

2 usages
@loanType.setter
def loanType(self, loanType):
    self.loanType = loanType

2 usages
@loanStatus.setter
def loanStatus(self, loanStatus):
    self.loanStatus = loanStatus
```

3. Home Loan Subclass:

```python
from Loan import Loan


2 usages
class HomeLoan(Loan):
    def __init__(self, loanID, customerID, principalAmount, interestRate, loanTerm, loanStatus, propertyAddress,
                 propertyValue):
        super().__init__(loanID, customerID, principalAmount, interestRate, loanTerm, loanType: "Home Loan", loanStatus)
        self.propertyAddress = propertyAddress
        self.propertyValue = propertyValue

    2 usages
    @property
    def propertyAddress(self):
        return self.propertyAddress

    2 usages
    @property
    def propertyValue(self):
        return self.propertyValue

    2 usages
    @propertyAddress.setter
    def propertyAddress(self, propertyAddress):
        self.propertyAddress = propertyAddress

    2 usages
    @propertyValue.setter
    def propertyValue(self, propertyValue):
        self.propertyValue = propertyValue
```

4. Car Loan Subclass:

```python
from Loan import Loan


2 usages
class CarLoan(Loan):
    def __init__(self, loanID, customerID, principalAmount, interestRate, loanTerm, loanStatus, carModel,
                 carValue):
        super().__init__(loanID, customerID, principalAmount, interestRate, loanTerm, loanType: "Car Loan", loanStatus)
        self.carModel = carModel
        self.carValue = carValue

    2 usages
    @property
    def carModel(self):
        return self.carModel

    2 usages
    @property
    def carValue(self):
        return self.carValue

    2 usages
    @carModel.setter
    def carModel(self, carModel):
        self.carModel = carModel

    2 usages
    @carValue.setter
    def carValue(self, carValue):
        self.carValue = carValue
```

## ABSTRACT CLASSES

1. ILoanRepository Class

```python
from abc import ABC, abstractmethod


class ILoanRepository(ABC):

    @abstractmethod
    def applyLoan(self):
        pass

    @abstractmethod
    def calculateInterest(self, loanID):
        pass

    @abstractmethod
    def loanStatus(self, loanID):
        pass

    @abstractmethod
    def calculateEMI(self, loanID):
        pass

    @abstractmethod
    def loanRepayment(self, loanID, amount):
        pass

    @abstractmethod
    def getAllLoan(self):
        pass

    @abstractmethod
    def getLoanById(self, loanID):
        pass
```

## IMPLEMENTATION CLASS

1. ILoanRepositoryImpl Class

```python
from Loan import Loan
from HomeLoan import HomeLoan
from CarLoan import CarLoan


2 usages
class ILoanRepositoryImpl:
    def __init__(self, dbUtil):
        self.dbUtil = dbUtil
```

```python
1 usage
def applyLoan(self):
    customerID = int(input("Enter the customerID: "))
    if self.getCustomerById(customerID) is None:
        raise Exception("CustomerNotFoundException")
    principalAmount = float(input("Enter the Principal Amount: "))
    inRate = float(input("Enter the Interest Rate: "))
    loanTerm = int(input("Enter the loan term in 'MONTHS': "))
    loanType = input("Enter the loan type('Car Loan', 'Home Loan', 'Other'). If 'Other', please mention the type "
                     "of loan e.g. 'Example Loan': ")
    loanStatus = "Pending"
    if loanType == "Home Loan":
        propertyAddress = input("Enter the property Address: ")
        propertyValue = float(input("Enter the property Value: "))
        loan = HomeLoan(self.generateUniqueLoanId(), customerID, principalAmount, inRate, loanTerm, loanStatus,
                        propertyAddress, propertyValue)
    elif loanType == "Car Loan":
        carModel = input("Enter the car model: ")
        carValue = float(input("Enter the car Value: "))
        loan = CarLoan(self.generateUniqueLoanId(), customerID, principalAmount, inRate, loanTerm, loanStatus,
                       carModel, carValue)
    else:
        loan = Loan(self.generateUniqueLoanId(), customerID, principalAmount, inRate, loanTerm, loanType,
                    loanStatus)

    confirm = input(print("Your loan ID has been created. Type Yes/No to confirm your Loan Application: "))
    if confirm.upper() == "YES":
        query = "Insert into loan values(%s, %s, %s, %s, %s, %s, %s)"
        values = (loan.loanID, loan.customerID, loan.principalAmount, loan.interestRate, loan.loanTerm,
                  loan.loanStatus, loan.loanType)
        self.dbUtil.executeQuery(query, values)
        return True
    else:
        return False
```

```python
def calculateInterest(self, loanID):
    query = "select principalAmount, interestRate, loanTermInMonths from loan where loanID = %s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    if result is None:
        raise Exception("InvalidLoanException")
    pAmount = result[0]
    inRate = result[1]
    term = result[2]
    interest = (pAmount * inRate * term) / 12
    return f"The interest for loanID {loanID} is Rs.{interest}"
```

```python
1 usage
def loanStatus(self, loanID):
    query = "Select creditScore from customer join loan on customer.customerID = loan.customerID and loanID = %s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    creditScore = result[0]
    if creditScore > 650:
        query1 = "update loan set loanStatus='Approved' where loanID=%s"
        values1 = (loanID,)
        self.dbUtil.executeQuery(query1, values1)
        return f"Loan Status Updated to 'Approved' for loanID {loanID}"
    else:
        query1 = "update loan set loanStatus='Rejected' where loanID=%s"
        values1 = (loanID,)
        self.dbUtil.executeQuery(query1, values1)
        return "Your CreditScore is too low. Loan Application Rejected!!!"
```

```python
1 usage
def calculateEMI(self, loanID):
    query = "select principalAmount, interestRate, loanTermInMonths from loan where loanID = %s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    if result is None:
        raise Exception("InvalidLoanException")
    pAmount = result[0]
    inRate = result[1]
    term = result[2]
    # EMI = [P * R * (1+R)^N] / [(1+R)^N-1]
    MIR = (inRate / 12) / 100
    x = (pAmount * MIR * ((1 + MIR) ** term))
    y = ((1 + MIR) ** term - 1)
    emi = x / y
    return f"EMI is calculated to be: {emi}"
```

```
1 usage
def loanRepayment(self, loanID, amount):
    emi_amt = self.calculateEMI(loanID)
    loanStatus = self.getLoanStatus(loanID)
    if loanStatus == "Approved":
        if amount<emi_amt:
            return f"Payment Rejected!!! Insufficient Amount as per the EMI. Your EMI amount: {emi_amt}"
        else:
            emiCount = amount//emi_amt
            balance = float(self.getPrincipalAmount(loanID)) - float(amount)
            query = "Update loan set principalAmount = %s where loanID=%s"
            values = (balance, loanID)
            self.dbUtil.executeQuery(query, values)
            emiCount -= 1
            return f"Payment Successfull. You can pay the rest in {emiCount} EMI's."
    elif loanStatus == "Pending":
        return f"You loan has not yet been approved. So please wait until its approved."
    else:
        return f"Your Loan has been rejected. Do not make any Payment."
```

```
1 usage
def getAllLoan(self):
    query = "select * from loan"
    result = self.dbUtil.fetchAll(query)
    for res in result:
        print(res)

1 usage
def getLoanById(self, loanID):
    query = "select * from loan where loanID=%s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    return result

1 usage
def get_no_of_loans(self):
    query = "select count(*) from loan"
    result = self.dbUtil.fetchOne(query)
    return result[0]

3 usages
def generateUniqueLoanId(self):
    return int(self.get_no_of_loans()) + 1

1 usage
def getCustomerById(self, customerID):
    query = "select * from customer where customerID = %s"
    values = (customerID,)
    result = self.dbUtil.fetchOne(query, values)
    return result

2 usages
def getLoanStatus(self, loanID):
    query = "select loanStatus from loan where loanID = %s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    return result[0]

1 usage
def getPrincipalAmount(self, loanID):
    query = "select principalAmount from loan where loanID = %s"
    values = (loanID,)
    result = self.dbUtil.fetchOne(query, values)
    return result[0]
```

## DATABASE CONNECTION CLASS

1. DB Util Class

```python
from mysql.connector import connect


2 usages
class DBUtil:
    def __init__(self, host, user, password, port, database):
        self.connection = connect(
            host=host,
            user=user,
            password=password,
            port=port,
            database=database
        )
        self.cursor = self.connection.cursor()

    def executeQuery(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            self.connection.commit()
        except Exception as e:
            print(f"Query Execution Error! {e}")
            self.connection.rollback()

    14 usages (14 dynamic)
    def fetchAll(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            return self.cursor.fetchall()
        except Exception as e:
            print(f"FetchAll Error: {e}")
            self.connection.rollback()

    def fetchOne(self, query, values=None):
        try:
            self.cursor.execute(query, values)
            return self.cursor.fetchone()
        except Exception as e:
            print(f"FetchOne Error! {e}")
            self.connection.rollback()

    1 usage
    def closeConnection(self):
        self.cursor.close()
        self.connection.close()
```

## MAIN APP

1. Loan Management System App:

```python
from DBUtil import DBUtil
from ILoanRepositoryImpl import ILoanRepositoryImpl


1 usage
class LoanManagement:
    def main(self):
        dbUtil = DBUtil(host='localhost', user='root', password='SQL@bunny11', port=3306, database='lmsystem')
        lrs = ILoanRepositoryImpl(dbUtil)
```

```python
        print("Welcome to the LOAN MANAGEMENT APP.")
        while True:
            print("What do you want to do?")
            print("1. Apply Loan")
            print("2. Get All Loans")
            print("3. Get Your Loan")
            print("4. Repay Your Loan")
            print("5. Check Your Loan Status")
            print("6. Update Your Loan Status")
            print("7. Calculate the Interest on your Loan")
            print("8. Exit")
            ch = int(input("Enter your choice: "))
            if ch == 1:
                result = lrs.applyLoan()
                if result:
                    print("Loan Applied")
                else:
                    print("Loan Application Cancelled")
            elif ch == 2:
                lrs.getAllLoan()
            elif ch == 3:
                loanID = int(input("Enter your loan ID: "))
                result = lrs.getLoanById(loanID)
                if result is None:
                    print("Invalid Loan ID!!!")
                else:
                    print(result)
            elif ch == 4:
                loanID = int(input("Enter your loan ID: "))
                amount = float(input("Enter the amount to repay: "))
                print(lrs.loanRepayment(loanID, amount))
            elif ch == 5:
                loanID = int(input("Enter your loan ID: "))
                print(lrs.getLoanStatus(loanID))
            elif ch == 6:
                loanID = int(input("Enter your loan ID: "))
                print(lrs.loanStatus(loanID))
            elif ch == 7:
                loanID = int(input("Enter your loan ID: "))
                print(lrs.calculateInterest(loanID))
            else:
                dbUtil.closeConnection()
                break
```

```python
if __name__ == '__main__':
    obj = LoanManagement()
    obj.main()
```

2. Output:

```
Welcome to the LOAN MANAGEMENT APP.
What do you want to do?
1. Apply Loan
2. Get All Loans
3. Get Your Loan
4. Repay Your Loan
5. Check Your Loan Status
6. Update Your Loan Status
7. Calculate the Interest on your Loan
8. Exit
Enter your choice: 2
(1, 1, Decimal('50000.00'), Decimal('5.50'), 36, 'Car Loan', 'Approved')
(2, 2, Decimal('100000.00'), Decimal('4.00'), 60, 'Home Loan', 'Rejected')
(3, 3, Decimal('75000.00'), Decimal('6.00'), 48, 'Car Loan', 'Pending')
(4, 4, Decimal('120000.00'), Decimal('3.50'), 72, 'Home Loan', 'Approved')
(5, 5, Decimal('60000.00'), Decimal('5.00'), 36, 'Car Loan', 'Pending')
(6, 6, Decimal('90000.00'), Decimal('4.50'), 60, 'Home Loan', 'Approved')
(7, 7, Decimal('80000.00'), Decimal('5.80'), 48, 'Home Loan', 'Approved')
(8, 8, Decimal('110000.00'), Decimal('3.20'), 72, 'Home Loan', 'Approved')
(9, 9, Decimal('95000.00'), Decimal('4.20'), 36, 'Car Loan', 'Approved')
(10, 10, Decimal('130000.00'), Decimal('3.80'), 60, 'Home Loan', 'Pending')
What do you want to do?
1. Apply Loan
2. Get All Loans
3. Get Your Loan
4. Repay Your Loan
5. Check Your Loan Status
6. Update Your Loan Status
7. Calculate the Interest on your Loan
8. Exit
Enter your choice: 8

Process finished with exit code 0
```