Coding Challenge

Loan Management System

```
import mysql.connector

class Customer:
    def __init__(self, customer_id=None, name=None, email=None, phone_number=None,
address=None, credit_score=None):
        self.customer_id = customer_id
        self.name = name
        self.email = email
        self.phone_number = phone_number
        self.address = address
        self.credit_score = credit_score

    def print_info(self):
        print("Customer ID:", self.customer_id)
        print("Name:", self.name)
        print("Email:", self.email)
        print("Phone Number:", self.phone_number)
        print("Address:", self.address)
        print("Credit Score:", self.credit_score)

    def get_customer_id(self):
        return self.customer_id

    def set_customer_id(self, customer_id):
        self.customer_id = customer_id

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_email(self):
        return self.email

    def set_email(self, email):
        self.email = email

    def get_phone_number(self):
        return self.phone_number

    def set_phone_number(self, phone_number):
        self.phone_number = phone_number

    def get_address(self):
        return self.address

    def set_address(self, address):
        self.address = address

    def get_credit_score(self):
        return self.credit_score
```

```
    def set_credit_score(self, credit_score):
        self.credit_score = credit_score


#from entity.Customer import Customer
class Loan:
    def __init__(self, loan_id=None, customer=None, principal_amount=None,
interest_rate=None, loan_term=None, loan_type=None, loan_status=None):
        self.loan_id = loan_id
        self.customer_id = self.customer_id
        self.principal_amount = principal_amount
        self.interest_rate = interest_rate
        self.loan_term = loan_term
        self.loan_type = loan_type
        self.loan_status = loan_status

    def print_info(self):
        print("Loan ID:", self.loan_id)
        print("Customer ID:", self.customer_id)
        print("Principal Amount:", self.principal_amount)
        print("Interest Rate:", self.interest_rate)
        print("Loan Term:", self.loan_term)
        print("Loan Type:", self.loan_type)
        print("Loan Status:", self.loan_status)

    def get_loan_id(self):
        return self.loan_id

    def set_loan_id(self, loan_id):
        self.loan_id = loan_id

    def set_customer_id(self, customer_id):
        self.customer_id = self.customer_id

    def get_principal_amount(self):
        return self.principal_amount

    def set_principal_amount(self, principal_amount):
        self.principal_amount = principal_amount

    def get_interest_rate(self):
        return self.interest_rate

    def set_interest_rate(self, interest_rate):
        self.interest_rate = interest_rate

    def get_loan_term(self):
        return self.loan_term

    def set_loan_term(self, loan_term):
        self.loan_term = loan_term

    def get_loan_type(self):
        return self.loan_type

    def set_loan_type(self, loan_type):
        self.loan_type = loan_type

    def get_loan_status(self):
        return self.loan_status
```

```
    def set_loan_status(self, loan_status):
        self.loan_status = loan_status

#from model.Loan import Loan
class HomeLoan(Loan):
    def __init__(self, loan_id=None, principal_amount=None, interest_rate=None,
loan_term=None, loan_status=None, property_address=None, property_value=None,
                 customer_id=None):
        super().__init__(loan_id, customer_id, principal_amount, interest_rate,
loan_term, "HomeLoan", loan_status)
        self.property_address = property_address
        self.property_value = property_value

    def print_info(self):
        super().print_info()
        print("Property Address:", self.property_address)
        print("Property Value:", self.property_value)

    def get_property_address(self):
        return self.property_address

    def set_property_address(self, property_address):
        self.property_address = property_address

    def get_property_value(self):
        return self.property_value

    def set_property_value(self, property_value):
        self.property_value = property_value


#from model.Loan import Loan
class CarLoan(Loan):
    def __init__(self, loan_id=None, customer_id=None, principal_amount=None,
interest_rate=None, loan_term=None, loan_status=None, car_model=None,
car_value=None):
        super().__init__(loan_id, customer_id, principal_amount, interest_rate,
loan_term, "CarLoan", loan_status)
        self.car_model = car_model
        self.car_value = car_value

    def print_info(self):
        super().print_info()
        print("Car Model:", self.car_model)
        print("Car Value:", self.car_value)

    def get_car_model(self):
        return self.car_model

    def set_car_model(self, car_model):
        self.car_model = car_model

    def get_car_value(self):
        return self.car_value

    def set_car_value(self, car_value):
        self.car_value = car_value


def connect_to_database():
    return mysql.connector.connect(
        host="localhost",
```

```
        user="root",
        password="HARSHA1@singh",
        database="LManage"
    )

def main():
    db_connection = connect_to_database()

    cursor = db_connection.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS Customer (customer_id INT PRIMARY KEY,
name VARCHAR(255), email VARCHAR(255), phone_number VARCHAR(15), address
VARCHAR(255), credit_score INT)")
    cursor.execute("CREATE TABLE IF NOT EXISTS Loan (loan_id INT PRIMARY KEY,
customer_id INT, principal_amount INT, interest_rate FLOAT, loan_term INT, loan_type
VARCHAR(50), loan_status VARCHAR(50))")
    cursor.execute("CREATE TABLE IF NOT EXISTS HomeLoan (loan_id INT PRIMARY KEY,
property_address VARCHAR(255), property_value INT)")
    cursor.execute("CREATE TABLE IF NOT EXISTS CarLoan (loan_id INT PRIMARY KEY,
car_model VARCHAR(255), car_value INT)")

    db_connection.commit()
    db_connection.close()

def insert_into_customer_table(customer_data):
    db_connection = connect_to_database()
    cursor = db_connection.cursor()

    sql = "INSERT INTO Customer (customer_id, name, email, phone_number, address,
credit_score) VALUES (%s, %s, %s, %s, %s, %s)"
    cursor.execute(sql, customer_data)

    db_connection.commit()
    db_connection.close()

def insert_into_loan_table(loan_data):
    db_connection = connect_to_database()
    cursor = db_connection.cursor()

    sql = "INSERT INTO Loan (loan_id, customer_id, principal_amount, interest_rate,
loan_term, loan_type, loan_status) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(sql, loan_data)

    db_connection.commit()
    db_connection.close()

def insert_into_home_loan_table(home_loan_data):
    db_connection = connect_to_database()
    cursor = db_connection.cursor()

    sql = "INSERT INTO HomeLoan (loan_id, property_address, property_value) VALUES
(%s, %s, %s)"
    cursor.execute(sql, home_loan_data)

    db_connection.commit()
    db_connection.close()

def insert_into_car_loan_table(car_loan_data):
    db_connection = connect_to_database()
    cursor = db_connection.cursor()

    sql = "INSERT INTO CarLoan (loan_id, car_model, car_value) VALUES (%s, %s, %s)"
    cursor.execute(sql, car_loan_data)
```

```
    db_connection.commit()
    db_connection.close()


if __name__ == "__main__":
    main()



from abc import ABC, abstractmethod
class ILoanRepository(ABC):
    @abstractmethod
    def applyLoan(self, loan):
        pass

    @abstractmethod
    def calculateInterest(self, loanId):
        pass

    @abstractmethod
    def calculateInterest(self, principal_amount, interest_rate, loan_term):
        pass

    @abstractmethod
    def loanStatus(self, loanId):
        pass

    @abstractmethod
    def calculateEMI(self, loanId):
        pass

    @abstractmethod
    def calculateEMI(self, principal_amount, interest_rate, loan_term):
        pass

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

    @abstractmethod
    def getAllLoan(self):
        pass

    @abstractmethod
    def getLoanById(self, loanId):
        pass

#from repository.iloan_repository import ILoanRepository
class ILoanRepositoryImpl(ILoanRepository):
    def __init__(self):
        self.db_connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="HARSHA1@singh",
            database="LManage"
        )



    def applyLoan(self, loan):
        try:
```

```
            cursor = self.db_connection.cursor()
            sql = "INSERT INTO Loan (loan_id, customer_id, principal_amount,
interest_rate, loan_term, loan_type, loan_status) VALUES (%s, %s, %s, %s, %s, %s,
%s)"
            cursor.execute(sql, (
            loan.loan_id, loan.customer_id, loan.principal_amount,
loan.interest_rate, loan.loan_term,
            loan.loan_type, "Pending"))
            self.db_connection.commit()
            cursor.close()
            print("Loan applied successfully.")
        except Exception as e:
            print("Error applying loan:", e)

    def calculateInterest(self, loanId):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT principal_amount, interest_rate, loan_term FROM Loan WHERE
loan_id = %s"
            cursor.execute(sql, (loanId,))
            loan_data = cursor.fetchone()
            if loan_data:
                principal_amount, interest_rate, loan_term = loan_data
                interest = (principal_amount * interest_rate * loan_term) / 12
                cursor.close()
                return interest
            else:
                raise Exception("Loan not found.")
        except Exception as e:
            print("Error calculating interest:", e)

    def calculateInterest(self, principal_amount, interest_rate, loan_term):
        return (principal_amount * interest_rate * loan_term) / 12

    def loanStatus(self, loanId):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT credit_score FROM Customer INNER JOIN Loan ON
Customer.customer_id = Loan.customer_id WHERE loan_id = %s"
            cursor.execute(sql, (loanId,))
            credit_score = cursor.fetchone()[0]
            if credit_score > 650:
                status = "Approved"
            else:
                status = "Rejected"
            sql_update = "UPDATE Loan SET loan_status = %s WHERE loan_id = %s"
            cursor.execute(sql_update, (status, loanId))
            self.db_connection.commit()
            cursor.close()
            print("Loan status updated successfully.")
        except Exception as e:
            print("Error updating loan status:", e)

    def calculateEMI(self, loanId):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT principal_amount, interest_rate, loan_term FROM Loan WHERE
loan_id = %s"
            cursor.execute(sql, (loanId,))
            loan_data = cursor.fetchone()
            if loan_data:
                principal_amount, interest_rate, loan_term = loan_data
```

```
                emi = (principal_amount * interest_rate * ((1 + interest_rate) **
loan_term)) / (
                            ((1 + interest_rate) ** loan_term) - 1)
                cursor.close()
                return emi
            else:
                raise Exception("Loan not found.")
        except Exception as e:
            print("Error calculating EMI:", e)

    def calculateEMI(self, principal_amount, interest_rate, loan_term):
        return (principal_amount * interest_rate * ((1 + interest_rate) **
loan_term)) / (
                    ((1 + interest_rate) ** loan_term) - 1)

    def loanRepayment(self, loanId, amount):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT principal_amount, interest_rate, loan_term FROM Loan WHERE
loan_id = %s"
            cursor.execute(sql, (loanId,))
            loan_data = cursor.fetchone()
            if loan_data:
                principal_amount, interest_rate, loan_term = loan_data
                emi = self.calculateEMI(principal_amount, interest_rate, loan_term)
                no_of_emi_paid = amount / emi
                if amount < emi:
                    print("Payment rejected. Amount is less than EMI.")
                else:
                    sql_update = "UPDATE Loan SET no_of_emi_paid = %s WHERE loan_id =
%s"
                    cursor.execute(sql_update, (no_of_emi_paid, loanId))
                    self.db_connection.commit()
                    print("Loan repayment successful.")
            else:
                raise Exception("Loan not found.")
        except Exception as e:
            print("Error in loan repayment:", e)

    def getAllLoan(self):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT * FROM Loan"
            cursor.execute(sql)
            loans = cursor.fetchall()
            for loan in loans:
                print(loan)
            cursor.close()
        except Exception as e:
            print("Error retrieving all loans:", e)

    def getLoanById(self, loanId):
        try:
            cursor = self.db_connection.cursor()
            sql = "SELECT * FROM Loan WHERE loan_id = %s"
            cursor.execute(sql, (loanId,))
            loan = cursor.fetchone()
            if loan:
                print(loan)
            else:
                raise Exception("Loan not found.")
            cursor.close()
```

```
        except Exception as e:
            print("Error retrieving loan by ID:", e)


# Example usage:
if __name__ == "__main__":
    loan_repo_impl = ILoanRepositoryImpl()

import mysql.connector

class DBUtil:
    @staticmethod
    def getDBConn():
        try:
            db_connection = mysql.connector.connect(
                host="localhost",
                user="root",
                password="HARSHA1@singh",
                database="Lmanage"
            )
            return db_connection
        except Exception as e:
            print("Error connecting to the database:", e)

if __name__ == "__main__":
    db_connection = DBUtil.getDBConn()
    if db_connection:
        print("Connection established successfully.")
    else:
        print("Failed to establish connection to the database.")

#import ILoanRepositoryImpl

#from util.db_util import DBUtil
class LoanManagement:
    def __init__(self):
        self.loan_repository = ILoanRepositoryImpl()

class DBUtil:
    @staticmethod
    def getDBConn():
        try:
            db_connection = mysql.connector.connect(
                host="localhost",
                user="root",
                password="HARSHA1@singh",
                database="Lmanage"
            )
            return db_connection
        except Exception as e:
            print("Error connecting to the database:", e)

class LoanManagement:
    def __init__(self):
        self.loan_repository = ILoanRepositoryImpl()

    def display_menu(self):
        print("\nLoan Management System")
        print("1. Apply for a loan")
        print("2. View all loans")
        print("3. View a specific loan")
        print("4. Make loan repayment")
```

```
        print("5. Exit")
        return input("Enter your choice: ")

    def apply_loan(self):
        try:
            customer_id = int(input("Enter customer ID: "))
            name = input("Enter customer name: ")
            email = input("Enter customer email: ")
            phone_number = input("Enter customer phone number: ")
            address = input("Enter customer address: ")
            credit_score = int(input("Enter customer credit score: "))

            customer = Customer(customer_id, name, email, phone_number, address,
credit_score)

            loan_id = int(input("Enter loan ID: "))
            principal_amount = float(input("Enter principal amount: "))
            interest_rate = float(input("Enter interest rate: "))
            loan_term = int(input("Enter loan term (in months): "))
            loan_type = input("Enter loan type (CarLoan or HomeLoan): ")


            if loan_type.lower() == "carloan":
                car_model = input("Enter car model: ")
                car_value = float(input("Enter car value: "))
                loan = CarLoan(loan_id, customer_id, principal_amount, interest_rate,
loan_term, "Pending", car_model,
                               car_value)
                sql = "INSERT INTO carloan (loan_id, car_model, car_value) VALUES
(%s, %s, %s)"
            elif loan_type.lower() == "homeloan":
                property_address = input("Enter property address: ")
                property_value = float(input("Enter property value: "))
                loan = HomeLoan(loan_id, customer_id, principal_amount,
interest_rate, loan_term, "Pending",
                                property_address, property_value)
            else:
                print("Invalid loan type. Please enter either 'CarLoan' or
'HomeLoan'.")
                return


            confirmation = input("Do you want to apply for this loan? (Yes/No): ")
            if confirmation.lower() == "yes":
                self.loan_repository.applyLoan(loan)
                print("Loan application successful.")
            else:
                print("Loan application canceled.")

        except Exception as e:
            print("Error applying for the loan:", e)
        pass

    def view_all_loans(self):
        print("\nAll Loans:")
        self.loan_repository.getAllLoan()

    def view_specific_loan(self):
        loan_id = int(input("\nEnter loan ID: "))
        self.loan_repository.getLoanById(loan_id)

    def make_loan_repayment(self):
```

```
        loan_id = int(input("\nEnter loan ID: "))
        amount = float(input("Enter repayment amount: "))
        self.loan_repository.loanRepayment(loan_id, amount)

    def main(self):
        while True:
            choice = self.display_menu()
            if choice == '1':
                self.apply_loan()
            elif choice == '2':
                self.view_all_loans()
            elif choice == '3':
                self.view_specific_loan()
            elif choice == '4':
                self.make_loan_repayment()
            elif choice == '5':
                print("\nExiting Loan Management System.")
                break
            else:
                print("\nInvalid choice. Please enter a valid option.")

if __name__ == "__main__":
    loan_management = LoanManagement()
    loan_management.main()
```