Coding Challenge
Meenakshi M
Loan Management

Entity Package:

```
class Customer:
   def __init__(self, customer_id=None, name="", email="", phone="",
address="", credit_score=0):
       self.customer_id = customer_id
       self.name = name
       self.email = email
       self.phone = phone
       self.address = address
       self.credit_score = credit_score

class Loan:
   def __init__(self, loan_id=None, customer=None, principal_amount=0.0,
interest_rate=0.0, loan_term=0, loan_type="", loan_status="Pending"):
       self.loan_id = loan_id
       self.customer = customer
       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
from LoanManagement.entity.loan import Loan

class CarLoan(Loan):
   def __init__(self, loan_id=None, customer=None, principal_amount=0.0,
interest_rate=0.0, loan_term=0, loan_status="Pending", car_model="",
car_value=0):
       super().__init__(loan_id, customer, principal_amount, interest_rate,
loan_term, loan_type="CarLoan", loan_status=loan_status)
       self.car_model = car_model
       self.car_value = car_value

from LoanManagement.entity.loan import Loan
class HomeLoan(Loan):
   def __init__(self, loan_id=None, customer=None, principal_amount=0.0,
interest_rate=0.0, loan_term=0, loan_status="Pending", property_address="",
property_value=0):
       super().__init__(loan_id, customer, principal_amount, interest_rate,
loan_term, loan_type="HomeLoan", loan_status=loan_status)
       self.property_address = property_address
       self.property_value = property_value
```

Doa Package:

```
from abc import ABC, abstractmethod
```

```python
from LoanManagement.entity.loan import Loan

class ILoanRepository(ABC):

    @abstractmethod
    def apply_loan(self, loan: Loan):
        pass

    @abstractmethod
    def calculate_interest(self, *args):
        pass

    @abstractmethod
    def loan_status(self, loan_id):
        pass

    @abstractmethod
    def calculate_emi(self, *args):
        pass

    @abstractmethod
    def loan_re_payment(self, loan_id, amount):
        pass

    @abstractmethod
    def get_all_loan(self):
        pass

    @abstractmethod
    def get_loan_by_id(self, loan_id):
        pass
```

```python
from LoanManagement.dao.ILoanRepository import ILoanRepository
from LoanManagement.util.db_conn_util import DBConnUtil
from LoanManagement.exceptions.invalid_loan_exception import
InvalidLoanException
import math

class LoanRepositoryImpl(ILoanRepository):
    def __init__(self):
        self.db = DBConnUtil()

    def apply_loan(self, loan):
        confirm = input("Do you want to apply for loan (Yes/No)? ").lower()
        if confirm != 'yes':
            print("Loan application cancelled.")
            return
        query = """
```

```
            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)
        """
        values = (loan.loan_id, loan.customer.customer_id,
loan.principal_amount, loan.interest_rate, loan.loan_term, loan.loan_type,
'Pending')
        self.db.execute_query(query, values)

    def calculate_interest(self, *args):
        if len(args) == 1:
            loan_id = args[0]
            result = self.db.fetch_query("SELECT principal_amount,
interest_rate, loan_term FROM loan WHERE loan_id = %s", (loan_id,))
            if not result:
                raise InvalidLoanException("Loan ID not found.")
            principal, rate, term = result[0]
        else:
            principal, rate, term = args
        interest = (principal * rate * term) / 12
        print(f"Interest Amount: {interest}")
        return interest

    def loan_status(self, loan_id):
        result = self.db.fetch_query("SELECT c.credit_score FROM loan l JOIN
customer c ON l.customer_id = c.customer_id WHERE l.loan_id = %s", (loan_id,))
        if not result:
            raise InvalidLoanException("Loan ID not found.")
        credit_score = result[0][0]
        status = 'Approved' if credit_score > 650 else 'Rejected'
        self.db.execute_query("UPDATE loan SET loan_status = %s WHERE loan_id =
%s", (status, loan_id))
        print(f"Loan {loan_id} is {status}.")

    def calculate_emi(self, *args):
        if len(args) == 1:
            loan_id = args[0]
            result = self.db.fetch_query(
                "SELECT principal_amount, interest_rate, loan_term FROM loan
WHERE loan_id = %s", (loan_id,))
            if not result:
                raise InvalidLoanException("Loan ID not found.")
            principal = float(result[0][0])
            rate = float(result[0][1])
            term = int(result[0][2])
            R = rate / 12 / 100
            EMI = (principal * R * pow(1 + R, term)) / (pow(1 + R, term) - 1)
            print(f"EMI: {round(EMI, 2)}")
        return EMI
```

```
    def loan_re_payment(self, loan_id, amount):
        emi = self.calculate_emi(loan_id)
        if amount < emi:
            print("Amount is less than EMI. Cannot process payment.")
            return
        num_emi_paid = math.floor(amount / emi)
        print(f"{num_emi_paid} EMI(s) paid for Loan ID {loan_id}.")
        # Extend: update DB for tracking paid EMIs

    def get_all_loan(self):
        results = self.db.fetch_query("SELECT * FROM loan")
        for row in results:
            print(row)

    def get_loan_by_id(self, loan_id):
        result = self.db.fetch_query("SELECT * FROM loan WHERE loan_id = %s",
(loan_id,))
        if not result:
            raise InvalidLoanException("Loan ID not found.")
        print(result[0])
```

Exception Package:

```
class InvalidLoanException(Exception):
    def __init__(self, message="Invalid Loan ID or Loan not found."):
        super().__init__(message)
```

Util Package:

```
import mysql.connector

class DBConnUtil:
    def __init__(self, host="localhost", user="root", password="root",
database="Loan"):
        self.conn = mysql.connector.connect(host=host, user=user,
password=password, database=database)
        self.cursor = self.conn.cursor()

    def execute_query(self, query, values=None):
        try:
            self.cursor.execute(query, values) if values else
self.cursor.execute(query)
            self.conn.commit()
            print("Successful!!!")
        except mysql.connector.Error as e:
            print(f"Error executing query: {e}")

    def fetch_query(self, query, values=None):
        try:
```

```
            self.cursor.execute(query, values) if values else
self.cursor.execute(query)
            result = self.cursor.fetchall()
            if result:
                print("Data retrieved successfully!")
            else:
                print("No records found.")
            return result
        except mysql.connector.Error as e:
            print(f"Error fetching data: {e}")
            return []

    def close_connection(self):
        self.cursor.close()
        self.conn.close()
```

Main package:

```
from LoanManagement.entity.customer import Customer
from LoanManagement.entity.loan import Loan
from LoanManagement.dao.ILoanRepositoryImpl import LoanRepositoryImpl
from LoanManagement.util.db_conn_util import DBConnUtil

def main():
    repo = LoanRepositoryImpl()
    db = DBConnUtil()

    while True:
        print("\n1. Apply for Loan")
        print("2. Get All Loans")
        print("3. Get Loan by ID")
        print("4. Loan Repayment")
        print("5. Calculate Interest")
        print("6. Calculate EMI")
        print("7. Loan Status")
        print("8. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            customer_id = int(input("Enter Customer ID: "))
            # Check if customer exists
            existing_customer = db.fetch_query("SELECT * FROM customer WHERE
customer_id = %s", (customer_id,))
            if not existing_customer:
                print("Customer not found. Please enter customer details.")
                name = input("Enter Name: ")
                email = input("Enter Email: ")
                phone = input("Enter Phone: ")
                address = input("Enter Address: ")
                credit_score = int(input("Enter Credit Score: "))
```

```
                customer = Customer(customer_id, name, email, phone, address,
credit_score)
                query = """
                    INSERT INTO customer (customer_id, name, email, phone,
address, credit_score)
                    VALUES (%s, %s, %s, %s, %s, %s)
                """
                db.execute_query(query, (customer.customer_id, customer.name,
customer.email, customer.phone, customer.address, customer.credit_score))
            else:
                customer = Customer(*existing_customer[0])

            loan_id = int(input("Enter Loan ID: "))
            principal = float(input("Enter Principal Amount: "))
            rate = float(input("Enter Interest Rate: "))
            term = int(input("Enter Loan Tenure (months): "))
            loan_type = input("Enter Loan Type (HomeLoan/CarLoan): ")
            loan = Loan(loan_id, customer, principal, rate, term, loan_type,
'Pending')

            repo.apply_loan(loan)

        elif choice == '2':
            repo.get_all_loan()

        elif choice == '3':
            loan_id = int(input("Enter Loan ID: "))
            repo.get_loan_by_id(loan_id)

        elif choice == '4':
            loan_id = int(input("Enter Loan ID: "))
            amount = float(input("Enter Repayment Amount: "))
            repo.loan_re_payment(loan_id, amount)

        elif choice == '5':
            loan_id = int(input("Enter Loan ID: "))
            repo.calculate_interest(loan_id)

        elif choice == '6':
            loan_id = int(input("Enter Loan ID: "))
            repo.calculate_emi(loan_id)

        elif choice == '7':
            loan_id = int(input("Enter Loan ID: "))
            repo.loan_status(loan_id)

        elif choice == '8':
            print("Exiting...")
            break
```

```
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    main()
```

Output:

```
Enter your choice: 1
Enter Customer ID: 11
No records found.
Customer not found. Please enter customer details.
Enter Name: Demi
Enter Email: demi.lovato@gmail.com
Enter Phone: 1234567890
Enter Address: dwndwndwnj
Enter Credit Score: 640
Successful!!!
Enter Loan ID: 1011
Enter Principal Amount: 500000
Enter Interest Rate: 6.5
Enter Loan Tenure (months): 14
Enter Loan Type (HomeLoan/CarLoan): HomeLoan
Do you want to apply for loan (Yes/No)? yes
Successful!!!
```

11	Demi	demi.lovato@gmail.com	1234567890	dwndwndwnj	640	
1011	11	500000.00	6.5	14	HomeLoan	Pending

```
Enter your choice: 2
Data retrieved successfully!
(1001, 1, Decimal('500000.00'), 7.5, 60, 'HomeLoan', 'Pending')
(1002, 2, Decimal('300000.00'), 8.0, 48, 'CarLoan', 'Pending')
(1003, 3, Decimal('400000.00'), 6.5, 36, 'HomeLoan', 'Pending')
(1004, 4, Decimal('250000.00'), 7.2, 24, 'CarLoan', 'Pending')
(1005, 5, Decimal('600000.00'), 7.8, 72, 'HomeLoan', 'Pending')
(1006, 6, Decimal('150000.00'), 8.5, 12, 'CarLoan', 'Pending')
(1007, 7, Decimal('200000.00'), 7.0, 18, 'CarLoan', 'Pending')
(1008, 8, Decimal('550000.00'), 6.9, 60, 'HomeLoan', 'Pending')
(1009, 9, Decimal('350000.00'), 8.1, 36, 'CarLoan', 'Pending')
(1010, 10, Decimal('450000.00'), 7.3, 48, 'HomeLoan', 'Pending')
(1011, 11, Decimal('500000.00'), 6.5, 14, 'HomeLoan', 'Pending')
```

```
Enter your choice: 3
Enter Loan ID: 1003
Data retrieved successfully!
(1003, 3, Decimal('400000.00'), 6.5, 36, 'HomeLoan', 'Pending')
```

```
Enter your choice: 4
Enter Loan ID: 1011
Enter Repayment Amount: 6000
Data retrieved successfully!
EMI: 37182.16
Amount is less than EMI. Cannot process payment.
```

```
Enter your choice: 6
Enter Loan ID: 1011
Data retrieved successfully!
EMI: 37182.16
```

```
Enter your choice: 5
Enter Loan ID: 1011
Data retrieved successfully!
Interest Amount: 3791666.6666666665
```