Coding Challenge - Loan Management System

Instructions

- Project submissions should be done through the partcipants' Github repository, and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive **Loan Management System** implemented with a strong focus on **SQL**, **control flow statements, loops, arrays, collections, exception handling**, **database interaction**.
- Follow **object-oriented principles** throughout the project. Use classes and objects to model real-world entities, **encapsulate data and behavior**, and **ensure code reusability**.
- Throw **user defined exceptions** from corresponding methods and handled**.**
- The following **Directory structure** is to be followed in the application.
    - **entity/model**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider interface to showcase functionalities.
        - Create the implementation class for the above interface with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a **DBPropertyUtil** class with a static function which takes property file name as parameter and returns connection string.
        - Create a **DBConnUtil** class which holds **static method** which takes connection string as parameter file and returns **connection object(Use method defined in DBPropertyUtil class to get the connection String)**.
- **main**
    - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Problem Statement:

Create SQL Schema from the customer and loan class, use the class attributes for table column names.

1. Define a **`Customer`** class with the following confidential attributes:
    a. Customer ID
    b. Name
    c. Email Address
    d. Phone Number
    e. Address
    f. creditScore
2. Define a base class **`Loan`** with the following attributes:
    a. loanId
    b. customer (reference of customer class)
    c. principalAmount

d. interestRate
e. loanTerm (Loan Tenure in months)
f. loanType (CarLoan, HomeLoan)
g. loanStatus (Pending, Approved)

3. Create two subclasses: **`HomeLoan`** and **`CarLoan`**. These subclasses should inherit from the **Loan** class and add attributes specific to their loan types. For example:
    a. **HomeLoan** should have a **propertyAddress** (String) and **propertyValue** (int) attribute.
    b. **CarLoan** should have a **carModel** (String) and **carValue** (int) attribute.
4. Implement the following for all classes.
    a. Write default constructors and overload the constructor with parameters, generate getter and setter, (print all information of attribute) methods for the attributes.
5. Define **ILoanRepository** interface/abstract class with following methods to interact with database.
    a. **applyLoan(loan Loan)**: pass appropriate parameters for creating loan. Initially loan status is pending and stored in database. before storing in database get confirmation from the user as Yes/No
    b. **calculateInterest(loanId)**: This method should calculate and return the interest amount for the loan. Loan should be retrieved from database and calculate the interest amount if loan not found generate **InvalidLoanException**.
        i. Overload the same method with required parameters to calculate the loan interest amount. It is used to calculate the loan interest while creating loan.
        ii. Interest = (Principal Amount * Interest Rate * Loan Tenure) / 12
    c. **loanStatus(loanId):** This method should display a message indicating that the loan is approved or rejected based on credit score, if credit score above 650 loan approved else rejected and should update in database.
    d. **calculateEMI(loanId)**: This method will calculate the emi amount for a month to repayment. Loan should be retrieved from database and calculate the interest amount, if loan not found generate **InvalidLoanException.**
        i. Overload the same method with required parameters to calculate the loan EMI amount. It is used to calculate the loan EMI while creating loan.
        ii. EMI = [P * R * (1+R)^N] / [(1+R)^N-1]
            1. EMI: The Equated Monthly Installment.
            2. P: Principal Amount (Loan Amount).
            3. R: Monthly Interest Rate (Annual Interest Rate / 12 / 100).
            4. N: Loan Tenure in months.
    e. **loanRepayment(loanId, amount):** calculate the noOfEmi can be paid from the amount if the amount is less than single emi reject the payment or pay the emi in whole number and update the variable.
    f. **getAllLoan():** get all loan as list and print the details.
    g. **getLoanById(loanId):** get loan and print the details, if loan not found generate **InvalidLoanException.**
6. Define **ILoanRepositoryImpl** class and implement the **ILoanRepository** interface and provide implementation of all methods.
7. Create **DBUtil** class and add the following method.

a. **static getDBConn():Connection** Establish a connection to the database and return Connection reference

8. Create **LoanManagement** main class and perform following operation:
    a. main method to simulate the loan management system. Allow the user to interact with the system by entering choice from menu such as "applyLoan", "getAllLoan", "getLoan", "loanRepayment", "exit."