Name : J209 Kishanth K

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."

Answers:

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

```
package com.java.loan.model;

public class Customer
{

   private int customerId;
   private String name;
   private String email;
   private String phoneNumber;
   private String address;
   private int creditScore;

        public int getCustomerId() {
                return customerId;
        }
        public void setCustomerId(int customerId) {
                this.customerId = customerId;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public String getPhoneNumber() {
                return phoneNumber;
        }
        public void setPhoneNumber(String phoneNumber) {
                this.phoneNumber = phoneNumber;
        }
        public String getAddress() {
                return address;
        }
```

```
	public void setAddress(String address) {
		this.address = address;
	}
	public int getCreditScore() {
		return creditScore;
	}
	public void setCreditScore(int creditScore) {
		this.creditScore = creditScore;
	}

	@Override
	public String toString()
	{
		return "Customer [customerId=" + customerId + ", name=" + name + ", email=" + email + ",
phoneNumber="
				+ phoneNumber + ", address=" + address + ", creditScore=" + creditScore +
"]";
	}

	public Customer(int customerId, String name, String email, String phoneNumber, String address,
int creditScore) {
		super();
		this.customerId = customerId;
		this.name = name;
		this.email = email;
		this.phoneNumber = phoneNumber;
		this.address = address;
		this.creditScore = creditScore;
	}

	public Customer() {
		super();
		// TODO Auto-generated constructor stub
	}



}
```

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)

```
package com.java.loan.model;

public abstract class Loan {
   private int loanId;
   private Customer customer;
   private double principalAmount;
   private double interestRate;
   private int loanTerm; // in months
   private LoanType loanType;
   private LoanStatus loanStatus;
   private double remainingAmount;
                                public int getLoanId() {
                                      return loanId;
                                }
                                public void setLoanId(int loanId) {
                                      this.loanId = loanId;
                                }
                                public Customer getCustomer() {
                                      return customer;
                                }
                                public void setCustomer(Customer customer) {
                                      this.customer = customer;
                                }
                                public double getPrincipalAmount() {
                                      return principalAmount;
                                }
                                public void setPrincipalAmount(double principalAmount) {
                                      this.principalAmount = principalAmount;
                                }
                                public double getInterestRate() {
                                      return interestRate;
                                }
                                public void setInterestRate(double interestRate) {
                                      this.interestRate = interestRate;
                                }
                                public int getLoanTerm() {
                                      return loanTerm;
                                }
                                public void setLoanTerm(int loanTerm) {
                                      this.loanTerm = loanTerm;
                                }
                                public LoanType getLoanType() {
                                      return loanType;
                                }
```

```
	public void setLoanType(LoanType loanType) {
		this.loanType = loanType;
	}
	public LoanStatus getLoanStatus() {
		return loanStatus;
	}
	public void setLoanStatus(LoanStatus loanStatus) {
		this.loanStatus = loanStatus;
	}
	public double getRemainingAmount() {
		return remainingAmount;
	}
	public void setRemainingAmount(double remainingAmount) {
		this.remainingAmount = remainingAmount;
	}

	@Override
	public String toString() {
		return "Loan [loanId=" + loanId + ", customer=" + customer + ",
principalAmount=" + principalAmount
				+ ", interestRate=" + interestRate + ", loanTerm=" +
loanTerm + ", remainingAmount=" + remainingAmount
				+ "]";
	}

	public Loan(int loanId, Customer customer, double principalAmount, double
interestRate, int loanTerm,
			LoanType loanType, LoanStatus loanStatus, double
remainingAmount) {
		super();
		this.loanId = loanId;
		this.customer = customer;
		this.principalAmount = principalAmount;
		this.interestRate = interestRate;
		this.loanTerm = loanTerm;
		this.loanType = loanType;
		this.loanStatus = loanStatus;
		this.remainingAmount = remainingAmount;
	}

	public Loan() {
		super();
		this.loanStatus = LoanStatus.PENDING;
		// TODO Auto-generated constructor stub
	}



}
```

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.

```
package com.java.loan.model;

public class HomeLoan extends Loan
{
   private String propertyAddress;
   private double propertyValue;

                                        public String getPropertyAddress() {
                                                return propertyAddress;
                                        }
                                        public void setPropertyAddress(String propertyAddress) {
                                                this.propertyAddress = propertyAddress;
                                        }
                                        public double getPropertyValue() {
                                                return propertyValue;
                                        }
                                        public void setPropertyValue(double propertyValue) {
                                                this.propertyValue = propertyValue;
                                        }

                                        @Override
                                        public String toString() {
                                                return "HomeLoan [propertyAddress=" + propertyAddress
+ ", propertyValue=" + propertyValue + "]";
                                        }

                                        public HomeLoan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
                                                        LoanType loanType, LoanStatus loanStatus, double
remainingAmount, String propertyAddress,
                                                        double propertyValue) {
                                                super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
                                                this.propertyAddress = propertyAddress;
                                                this.propertyValue = propertyValue;
                                        }

                                        public HomeLoan() {
                                                super();
                                                setLoanType(LoanType.HOME);
                                                // TODO Auto-generated constructor stub
                                        }

                                        public HomeLoan(int loanId, Customer customer, double
```

```
principalAmount, double interestRate, int loanTerm,
                                                    LoanType loanType, LoanStatus loanStatus, double
remainingAmount) {
                                                super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
                                                // TODO Auto-generated constructor stub
                                            }


}
```

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.

Customer.java

```
package com.java.loan.model;

public class Customer
{

                                        private int customerId;
                                        private String name;
                                        private String email;
                                        private String phoneNumber;
                                        private String address;
                                        private int creditScore;

                                        public int getCustomerId() {
                                                return customerId;
                                        }
                                        public void setCustomerId(int customerId) {
                                                this.customerId = customerId;
                                        }
                                        public String getName() {
                                                return name;
                                        }
                                        public void setName(String name) {
                                                this.name = name;
                                        }
                                        public String getEmail() {
                                                return email;
                                        }
                                        public void setEmail(String email) {
                                                this.email = email;
```

```
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public int getCreditScore() {
		return creditScore;
	}
	public void setCreditScore(int creditScore) {
		this.creditScore = creditScore;
	}

	@Override
	public String toString()
	{
		return "Customer [customerId=" + customerId + ", name="
+ name + ", email=" + email + ", phoneNumber="
				+ phoneNumber + ", address=" + address
+ ", creditScore=" + creditScore + "]";
	}

	public Customer(int customerId, String name, String email, String
phoneNumber, String address, int creditScore) {
		super();
		this.customerId = customerId;
		this.name = name;
		this.email = email;
		this.phoneNumber = phoneNumber;
		this.address = address;
		this.creditScore = creditScore;
	}

	public Customer() {
		super();
		// TODO Auto-generated constructor stub
	}



}
```

Loan.java

```
package com.java.loan.model;

public abstract class Loan {
                                        private int loanId;
                                        private Customer customer;
                                        private double principalAmount;
                                        private double interestRate;
                                        private int loanTerm; // in months
                                        private LoanType loanType;
                                        private LoanStatus loanStatus;
                                        private double remainingAmount;
                                        public int getLoanId() {
                                                return loanId;
                                        }
                                        public void setLoanId(int loanId) {
                                                this.loanId = loanId;
                                        }
                                        public Customer getCustomer() {
                                                return customer;
                                        }
                                        public void setCustomer(Customer customer) {
                                                this.customer = customer;
                                        }
                                        public double getPrincipalAmount() {
                                                return principalAmount;
                                        }
                                        public void setPrincipalAmount(double principalAmount) {
                                                this.principalAmount = principalAmount;
                                        }
                                        public double getInterestRate() {
                                                return interestRate;
                                        }
                                        public void setInterestRate(double interestRate) {
                                                this.interestRate = interestRate;
                                        }
                                        public int getLoanTerm() {
                                                return loanTerm;
                                        }
                                        public void setLoanTerm(int loanTerm) {
                                                this.loanTerm = loanTerm;
                                        }
                                        public LoanType getLoanType() {
                                                return loanType;
                                        }
                                        public void setLoanType(LoanType loanType) {
                                                this.loanType = loanType;
                                        }
                                        public LoanStatus getLoanStatus() {
                                                return loanStatus;
```

```
	}
	public void setLoanStatus(LoanStatus loanStatus) {
		this.loanStatus = loanStatus;
	}
	public double getRemainingAmount() {
		return remainingAmount;
	}
	public void setRemainingAmount(double remainingAmount) {
		this.remainingAmount = remainingAmount;
	}

	@Override
	public String toString() {
		return "Loan [loanId=" + loanId + ", customer=" +
customer + ", principalAmount=" + principalAmount
				+ ", interestRate=" + interestRate + ",
loanTerm=" + loanTerm + ", remainingAmount=" + remainingAmount
				+ "]";
	}

	public Loan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
			LoanType loanType, LoanStatus loanStatus, double
remainingAmount) {
		super();
		this.loanId = loanId;
		this.customer = customer;
		this.principalAmount = principalAmount;
		this.interestRate = interestRate;
		this.loanTerm = loanTerm;
		this.loanType = loanType;
		this.loanStatus = loanStatus;
		this.remainingAmount = remainingAmount;
	}

	public Loan() {
		super();
		this.loanStatus = LoanStatus.PENDING;
		// TODO Auto-generated constructor stub
	}


}
```

HomeLoan.java

```
package com.java.loan.model;

public class HomeLoan extends Loan
{
                                        private String propertyAddress;
                                        private double propertyValue;

                                        public String getPropertyAddress() {
                                                return propertyAddress;
                                        }
                                        public void setPropertyAddress(String propertyAddress) {
                                                this.propertyAddress = propertyAddress;
                                        }
                                        public double getPropertyValue() {
                                                return propertyValue;
                                        }
                                        public void setPropertyValue(double propertyValue) {
                                                this.propertyValue = propertyValue;
                                        }

                                        @Override
                                        public String toString() {
                                                return "HomeLoan [propertyAddress=" + propertyAddress
+ ", propertyValue=" + propertyValue + "]";
                                        }

                                        public HomeLoan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
                                                        LoanType loanType, LoanStatus loanStatus, double
remainingAmount, String propertyAddress,
                                                        double propertyValue) {
                                                super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
                                                this.propertyAddress = propertyAddress;
                                                this.propertyValue = propertyValue;
                                        }

                                        public HomeLoan() {
                                                super();
                                                setLoanType(LoanType.HOME);
                                                // TODO Auto-generated constructor stub
                                        }

                                        public HomeLoan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
                                                        LoanType loanType, LoanStatus loanStatus, double
remainingAmount) {
                                                super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
```

```
                                        // TODO Auto-generated constructor stub
                                }


}
```

CarLoan.java

```
package com.java.loan.model;

public class CarLoan extends Loan
{
                                private String carModel;
                                private double carValue;

                                public String getCarModel() {
                                        return carModel;
                                }
                                public void setCarModel(String carModel) {
                                        this.carModel = carModel;
                                }
                                public double getCarValue() {
                                        return carValue;
                                }
                                public void setCarValue(double carValue) {
                                        this.carValue = carValue;
                                }

                                @Override
                                public String toString() {
                                        return "CarLoan [carModel=" + carModel + ", carValue=" +
carValue + "]";
                                }

                                public CarLoan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
                                                LoanType loanType, LoanStatus loanStatus, double
remainingAmount, String carModel, double carValue) {
                                        super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
                                        this.carModel = carModel;
                                        this.carValue = carValue;
                                }

                                public CarLoan() {
                                        super();
                                        setLoanType(LoanType.CAR);
                                        // TODO Auto-generated constructor stub
                                }
```

```
                                        public CarLoan(int loanId, Customer customer, double
principalAmount, double interestRate, int loanTerm,
                                                LoanType loanType, LoanStatus loanStatus, double
remainingAmount) {
                                                super(loanId, customer, principalAmount, interestRate,
loanTerm, loanType, loanStatus, remainingAmount);
                                                // TODO Auto-generated constructor stub
                                        }


}
```

LoanStatus.java

```
package com.java.loan.model;

public enum LoanStatus {
                                        ENDING, APPROVED, REJECTED, PENDING

}
```

LoanType.java

```
package com.java.loan.model;

public enum LoanType {
                                        CAR, HOME
}
```

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.

ILoanRepository.java

```
package com.java.loan.dao;

import com.java.loan.exception.InvalidLoanException;
import com.java.loan.model.Loan;
import java.util.List;

public interface ILoanRepository {
	boolean applyLoan(Loan loan);

	double calculateInterest(int loanId) throws InvalidLoanException;
	double calculateInterest(double principal, double rate, int term);

	boolean loanStatus(int loanId) throws InvalidLoanException;

	double calculateEMI(int loanId) throws InvalidLoanException;
	double calculateEMI(double principal, double rate, int term);

	boolean loanRepayment(int loanId, double amount) throws InvalidLoanException;

	List<Loan> getAllLoan();

	Loan getLoanById(int loanId) throws InvalidLoanException;
}
```

InvalidLoanException.java

```
package com.java.loan.exception;

public class InvalidLoanException extends Exception {
	public InvalidLoanException(String message) {
		super(message);
	}
}
```

6. Define **ILoanRepositoryImpl** class and implement the **ILoanRepository** interface and provide implementation of all methods.

ILoanRepositoryImpl.java

```
package com.java.loan.dao.impl;

import com.java.loan.dao.ILoanRepository;
import com.java.loan.exception.InvalidLoanException;
import com.java.loan.util.DBConnUtil;
import com.java.loan.model.*;
import com.java.loan.util.DBPropertyUtil;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class ILoanRepositoryImpl implements ILoanRepository {
                                        private Connection getConnection() throws SQLException {
                                         String connStr =
DBPropertyUtil.getConnectionString("db.properties");
                                         return DBConnUtil.getConnection(connStr);
                                        }

                                        private int insertCustomer(Customer customer) throws
SQLException {
                                         String sql = "INSERT INTO Customer (name, email, phone_number,
address, credit_score) " +
                                                 "VALUES (?, ?, ?, ?, ?)";

                                         try (Connection conn = getConnection();
                                                 PreparedStatement pstmt =
conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {

                                                 pstmt.setString(1, customer.getName());
                                                 pstmt.setString(2, customer.getEmail());
                                                 pstmt.setString(3, customer.getPhoneNumber());
                                                 pstmt.setString(4, customer.getAddress());
                                                 pstmt.setInt(5, customer.getCreditScore());

                                                 int affectedRows = pstmt.executeUpdate();
                                                 if (affectedRows == 0) {
                                                         throw new SQLException("Creating customer
failed, no rows affected.");
                                                 }

                                                 try (ResultSet generatedKeys = pstmt.getGeneratedKeys())
```

```
{
                                                    if (generatedKeys.next()) {
                                                            int customerId = generatedKeys.getInt(1);
                                                            customer.setCustomerId(customerId);
                                                            return customerId;
                                                    } else {
                                                            throw new SQLException("Creating
customer failed, no ID obtained.");
                                                    }
                                            }
                                    }
                                    }

                                    @Override
                                    public boolean applyLoan(Loan loan) {
                                     Scanner scanner = new Scanner(System.in);
                                     System.out.print("Confirm loan application (Yes/No): ");
                                     if (!scanner.nextLine().equalsIgnoreCase("yes")) {
                                            return false;
                                     }

                                     try {

                                            int customerId = insertCustomer(loan.getCustomer());

                                            String sql = "INSERT INTO Loan (customer_id,
principal_amount, interest_rate, loan_term, loan_type, loan_status, remaining_amount) " +
                                                    "VALUES (?, ?, ?, ?, ?, ?, ?)";

                                            try (Connection conn = getConnection();
                                                    PreparedStatement pstmt =
conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {

                                                    pstmt.setInt(1, customerId);
                                                    pstmt.setDouble(2, loan.getPrincipalAmount());
                                                    pstmt.setDouble(3, loan.getInterestRate());
                                                    pstmt.setInt(4, loan.getLoanTerm());
                                                    pstmt.setString(5, loan.getLoanType().toString());
                                                    pstmt.setString(6,
LoanStatus.PENDING.toString());
                                                    pstmt.setDouble(7, loan.getPrincipalAmount());

                                                    int affectedRows = pstmt.executeUpdate();
                                                    if (affectedRows == 0) {
                                                            return false;
                                                    }

                                                    try (ResultSet generatedKeys =
pstmt.getGeneratedKeys()) {

                                                            if (generatedKeys.next()) {
                                                                    int loanId =
```

```
generatedKeys.getInt(1);
                                                        loan.setLoanId(loanId);

                                                        if (loan instanceof HomeLoan) {
                                                                insertHomeLoan(loanId,
(HomeLoan) loan);
                                                        } else if (loan instanceof CarLoan) {
                                                                insertCarLoan(loanId,
(CarLoan) loan);
                                                        }
                                                        return true;
                                                }
                                        }
                                }
                                return false;
                        } catch (SQLException e) {
                                e.printStackTrace();
                                return false;
                        }
                }

                private void insertHomeLoan(int loanId, HomeLoan loan) throws
SQLException {
                        String sql = "INSERT INTO HomeLoan (loan_id, property_address,
property_value) VALUES (?, ?, ?)";
                        try (Connection conn = getConnection();
                                        PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                                pstmt.setInt(1, loanId);
                                pstmt.setString(2, loan.getPropertyAddress());
                                pstmt.setDouble(3, loan.getPropertyValue());
                                pstmt.executeUpdate();
                        }
                }

                private void insertCarLoan(int loanId, CarLoan loan) throws
SQLException {
                        String sql = "INSERT INTO CarLoan (loan_id, car_model, car_value)
VALUES (?, ?, ?)";
                        try (Connection conn = getConnection();
                                        PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                                pstmt.setInt(1, loanId);
                                pstmt.setString(2, loan.getCarModel());
                                pstmt.setDouble(3, loan.getCarValue());
                                pstmt.executeUpdate();
                        }
                }

                @Override
                public double calculateInterest(int loanId) throws
```

```
InvalidLoanException {
                                        Loan loan = getLoanById(loanId);
                                        if (loan == null) {
                                                throw new InvalidLoanException("Loan not found with ID:
" + loanId);
                                        }
                                        return calculateInterest(loan.getPrincipalAmount(),
loan.getInterestRate(), loan.getLoanTerm());
                                        }

                                        @Override
                                        public double calculateInterest(double principal, double rate, int
term) {
                                        return (principal * rate * term) / 1200;
                                        }

                                        @Override
                                        public boolean loanStatus(int loanId) throws InvalidLoanException {
                                        Loan loan = getLoanById(loanId);
                                        if (loan == null) {
                                                throw new InvalidLoanException("Loan not found with ID:
" + loanId);
                                        }

                                        boolean isApproved = loan.getCustomer().getCreditScore() > 650;
                                        LoanStatus newStatus = isApproved ? LoanStatus.APPROVED :
LoanStatus.REJECTED;

                                        String sql = "UPDATE Loan SET loan_status = ? WHERE loan_id =
?";
                                        try (Connection conn = getConnection();
                                                        PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                                                pstmt.setString(1, newStatus.toString());
                                                pstmt.setInt(2, loanId);
                                                return pstmt.executeUpdate() > 0;
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                                return false;
                                        }
                                        }

                                        @Override
                                        public double calculateEMI(int loanId) throws InvalidLoanException
{
                                        Loan loan = getLoanById(loanId);
                                        if (loan == null) {
                                                throw new InvalidLoanException("Loan not found with ID:
" + loanId);
                                        }
                                        return calculateEMI(loan.getPrincipalAmount(),
```

```
loan.getInterestRate(), loan.getLoanTerm());
                                        }

                                        @Override
                                        public double calculateEMI(double principal, double rate, int term)
{
                                         double monthlyRate = rate / 1200; // Convert annual rate to
monthly
                                         return (principal * monthlyRate * Math.pow(1 + monthlyRate,
term)) /
                                                        (Math.pow(1 + monthlyRate, term) - 1);
                                        }

                                        @Override
                                        public boolean loanRepayment(int loanId, double amount) throws
InvalidLoanException {
                                         Loan loan = getLoanById(loanId);
                                         if (loan == null) {
                                                throw new InvalidLoanException("Loan not found with ID:
" + loanId);
                                         }

                                         double emi = calculateEMI(loanId);
                                         if (amount < emi) {
                                                return false;
                                         }

                                         int numberOfEmis = (int) (amount / emi);
                                         double newRemainingAmount = loan.getRemainingAmount() -
(emi * numberOfEmis);

                                         String sql = "UPDATE Loan SET remaining_amount = ? WHERE
loan_id = ?";
                                         try (Connection conn = getConnection();
                                                        PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                                                pstmt.setDouble(1, newRemainingAmount);
                                                pstmt.setInt(2, loanId);
                                                return pstmt.executeUpdate() > 0;
                                         } catch (SQLException e) {
                                                e.printStackTrace();
                                                return false;
                                         }
                                        }

                                        @Override
                                        public List<Loan> getAllLoan() {
                                         List<Loan> loans = new ArrayList<>();
                                         String sql = "SELECT l.*, c.*, " +
                                                        "h.property_address, h.property_value, " +
                                                        "ca.car_model, ca.car_value " +
```

```
                                                "FROM Loan l " +
                                                "JOIN Customer c ON l.customer_id =
c.customer_id " +
                                                "LEFT JOIN HomeLoan h ON l.loan_id = h.loan_id "
+
                                                "LEFT JOIN CarLoan ca ON l.loan_id = ca.loan_id";

                                        try (Connection conn = getConnection();
                                                Statement stmt = conn.createStatement();
                                                ResultSet rs = stmt.executeQuery(sql)) {

                                                while (rs.next()) {
                                                        loans.add(createLoanFromResultSet(rs));
                                                }
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                        }
                                        return loans;
                                    }

                                    @Override
                                    public Loan getLoanById(int loanId) throws InvalidLoanException {
                                        String sql = "SELECT l.*, c.*, " +
                                                "h.property_address, h.property_value, " +
                                                "ca.car_model, ca.car_value " +
                                                "FROM Loan l " +
                                                "JOIN Customer c ON l.customer_id =
c.customer_id " +
                                                "LEFT JOIN HomeLoan h ON l.loan_id = h.loan_id "
+
                                                "LEFT JOIN CarLoan ca ON l.loan_id = ca.loan_id "
+
                                                "WHERE l.loan_id = ?";

                                        try (Connection conn = getConnection();
                                                PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                                                pstmt.setInt(1, loanId);
                                                try (ResultSet rs = pstmt.executeQuery()) {
                                                        if (rs.next()) {
                                                                return createLoanFromResultSet(rs);
                                                        }
                                                }
                                        } catch (SQLException e) {
                                                e.printStackTrace();
                                        }
                                        throw new InvalidLoanException("Loan not found with ID: " +
loanId);
                                    }

                                    private Loan createLoanFromResultSet(ResultSet rs) throws
```

```
SQLException {
                                        Customer customer = new Customer(
                                                rs.getInt("customer_id"),
                                                rs.getString("name"),
                                                rs.getString("email"),
                                                rs.getString("phone_number"),
                                                rs.getString("address"),
                                                rs.getInt("credit_score")
                                                );

                                        LoanType loanType = LoanType.valueOf(rs.getString("loan_type"));
                                        Loan loan;

                                        if (loanType == LoanType.HOME) {
                                                loan = new HomeLoan();
                                                ((HomeLoan)
loan).setPropertyAddress(rs.getString("property_address"));
                                                ((HomeLoan)
loan).setPropertyValue(rs.getDouble("property_value"));
                                        } else {
                                                loan = new CarLoan();
                                                ((CarLoan) loan).setCarModel(rs.getString("car_model"));
                                                ((CarLoan) loan).setCarValue(rs.getDouble("car_value"));
                                        }

                                        loan.setLoanId(rs.getInt("loan_id"));
                                        loan.setCustomer(customer);
                                        loan.setPrincipalAmount(rs.getDouble("principal_amount"));
                                        loan.setInterestRate(rs.getDouble("interest_rate"));
                                        loan.setLoanTerm(rs.getInt("loan_term"));

                                       loan.setLoanStatus(LoanStatus.valueOf(rs.getString("loan_status"))
);
                                        loan.setRemainingAmount(rs.getDouble("remaining_amount"));

                                        return loan;
                                       }
}
```

7. Create **DBUtil** class and add the following method.
    a. **static getDBConn():Connection** Establish a connection to the database and return Connection reference

DBConnUtil.java

```
package com.java.loan.util;

import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.SQLException;

public class DBConnUtil {
	public static Connection getConnection(String connectionString)
throws SQLException {
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			return DriverManager.getConnection(connectionString,
"root", "327748");
		} catch (ClassNotFoundException e) {
			throw new SQLException("Database driver not found", e);
		}
	}
}
```

DBPropertyUtil.java

```
package com.java.loan.util;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class DBPropertyUtil {
	public static String getConnectionString(String propertyFileName) {
		Properties properties = new Properties();
		try (InputStream input =
DBPropertyUtil.class.getClassLoader().getResourceAsStream(propertyFileName)) {
			if (input == null) {
				throw new RuntimeException("Unable to find " +
propertyFileName);
			}
			properties.load(input);

			return properties.getProperty("url");
		} catch (IOException e) {
			throw new RuntimeException("Error loading database
properties", e);
		}
	}
}
```

GitHub : https://github.com/Kishanth1117/CodingChallenge2-LoanManagementSystem.git