JAVA CODING CHALLENGE

Name: Kavin Kaarthik Date: 08/04/2025  
Superset Id: 5371276 Batch: 3

SQL queries used to create table:   
  
use loan\_management;

CREATE TABLE Customer (

customerId INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

emailAddress VARCHAR(100) UNIQUE NOT NULL,

phoneNumber VARCHAR(15) UNIQUE NOT NULL,

address VARCHAR(255),

creditScore INT

);

CREATE TABLE Loan (

loanId INT PRIMARY KEY,

customerId INT,

principalAmount DECIMAL(15, 2),

interestRate DECIMAL(5, 2),

loanTerm INT,

loanType VARCHAR(50),

loanStatus VARCHAR(20),

FOREIGN KEY (customerId) REFERENCES Customer(customerId)

);

show tables;  
  
java coding part:   
  
task 1- 4 : creating classes and getter/setters for the attributes   
  
package entity;

public class Customer {

private int customerId;

private String name;

private String emailAddress;

private String phoneNumber;

private String address;

private int creditScore;

public Customer() {}

public Customer(int customerId, String name, String emailAddress, String phoneNumber, String address, int creditScore) {

this.customerId = customerId;

this.name = name;

this.emailAddress = emailAddress;

this.phoneNumber = phoneNumber;

this.address = address;

this.creditScore = 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 getEmailAddress() {

return emailAddress;

}

public void setEmailAddress(String emailAddress) {

this.emailAddress = emailAddress;

}

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;

}

public void printDetails() {

System.***out***.println("----- Customer Details -----");

System.***out***.println("Customer ID: " + customerId);

System.***out***.println("Name: " + name);

System.***out***.println("Email Address: " + emailAddress);

System.***out***.println("Phone Number: " + phoneNumber);

System.***out***.println("Address: " + address);

System.***out***.println("Credit Score: " + creditScore);

}

}

Loan class:   
  
package entity;

public class Loan {

private int loanId;

private Customer customer;

private double principalAmount;

private double interestRate;

private int loanTerm;

private String loanType;

private String loanStatus;

public Loan() {}

public Loan(int loanId, Customer customer, double principalAmount, double interestRate,

int loanTerm, String loanType, String loanStatus) {

this.loanId = loanId;

this.customer = customer;

this.principalAmount = principalAmount;

this.interestRate = interestRate;

this.loanTerm = loanTerm;

this.loanType = loanType;

this.loanStatus = loanStatus;

}

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 String getLoanType() {

return loanType;

}

public void setLoanType(String loanType) {

this.loanType = loanType;

}

public String getLoanStatus() {

return loanStatus;

}

public void setLoanStatus(String loanStatus) {

this.loanStatus = loanStatus;

}

public void printDetails() {

System.***out***.println("----- Loan Details -----");

System.***out***.println("Loan ID: " + loanId);

System.***out***.println("Customer ID: " + customer.getCustomerId());

System.***out***.println("Principal Amount: " + principalAmount);

System.***out***.println("Interest Rate: " + interestRate);

System.***out***.println("Loan Term (in months): " + loanTerm);

System.***out***.println("Loan Type: " + loanType);

System.***out***.println("Loan Status: " + loanStatus);

}

}

home loan class:   
  
package entity;

public class HomeLoan extends Loan {

private String propertyAddress;

private int propertyValue;

public HomeLoan() {

super();

}

public HomeLoan(int loanId, Customer customer, double principalAmount, double interestRate,

int loanTerm, String loanType, String loanStatus,

String propertyAddress, int propertyValue) {

super(loanId, customer, principalAmount, interestRate, loanTerm, loanType, loanStatus);

this.propertyAddress = propertyAddress;

this.propertyValue = propertyValue;

}

public String getPropertyAddress() {

return propertyAddress;

}

public void setPropertyAddress(String propertyAddress) {

this.propertyAddress = propertyAddress;

}

public int getPropertyValue() {

return propertyValue;

}

public void setPropertyValue(int propertyValue) {

this.propertyValue = propertyValue;

}

*@Override*

public void printDetails() {

super.printDetails();

System.***out***.println("----- Home Loan Extra Details -----");

System.***out***.println("Property Address: " + propertyAddress);

System.***out***.println("Property Value: " + propertyValue);

}

}

CarLoan class:

package entity;

public class CarLoan extends Loan {

private String carModel;

private int carValue;

public CarLoan() {

super();

}

public CarLoan(int loanId, Customer customer, double principalAmount, double interestRate,

int loanTerm, String loanType, String loanStatus,

String carModel, int carValue) {

super(loanId, customer, principalAmount, interestRate, loanTerm, loanType, loanStatus);

this.carModel = carModel;

this.carValue = carValue;

}

public String getCarModel() {

return carModel;

}

public void setCarModel(String carModel) {

this.carModel = carModel;

}

public int getCarValue() {

return carValue;

}

public void setCarValue(int carValue) {

this.carValue = carValue;

}

*@Override*

public void printDetails() {

super.printDetails();

System.***out***.println("----- Car Loan Extra Details -----");

System.***out***.println("Car Model: " + carModel);

System.***out***.println("Car Value: " + carValue);

}

}

Task 5-6: ILoanRepository interface/abstract class and Define ILoanRepositoryImpl class and implement the ILoanRepository interface and provide implementation of all methods.

package dao;

import exception.InvalidLoanException;

import entity.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 tenureMonths);

void loanStatus(int loanId) throws InvalidLoanException;

double calculateEMI(int loanId) throws InvalidLoanException;

double calculateEMI(double principal, double rate, int tenureMonths);

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

List<Loan> getAllLoan();

Loan getLoanById(int loanId) throws InvalidLoanException;

}

Class LoanRepositoryImpl:   
  
package dao;

import entity.\*;

import exception.InvalidLoanException;

import util.DBUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class LoanRepositoryImpl implements ILoanRepository {

private Connection connection;

public LoanRepositoryImpl() {

this.connection = DBUtil.*getDBConn*();

}

*@Override*

public boolean applyLoan(Loan loan) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Do you want to proceed with applying for the loan? (Yes/No): ");

String confirmation = scanner.nextLine();

if (!confirmation.equalsIgnoreCase("Yes")) {

System.***out***.println("Loan application cancelled.");

return false;

}

try {

String sql = "INSERT INTO Loan (loanId, customerId, principalAmount, interestRate, loanTerm, loanType, loanStatus) " +

"VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, loan.getLoanId());

statement.setInt(2, loan.getCustomer().getCustomerId());

statement.setDouble(3, loan.getPrincipalAmount());

statement.setDouble(4, loan.getInterestRate());

statement.setInt(5, loan.getLoanTerm());

statement.setString(6, loan.getLoanType());

statement.setString(7, "Pending"); // Default status

int rowsInserted = statement.executeUpdate();

if (rowsInserted > 0) {

System.***out***.println("Loan application submitted successfully.");

return true;

} else {

System.***out***.println("Loan application failed.");

return false;

}

} catch (SQLException e) {

System.***out***.println("Error while applying for loan: " + e.getMessage());

return false;

}

}

*@Override*

public double calculateInterest(int loanId) throws InvalidLoanException {

String sql = "SELECT principalAmount, interestRate, loanTerm FROM Loan WHERE loanId = ?";

try (PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setInt(1, loanId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

double principal = resultSet.getDouble("principalAmount");

double rate = resultSet.getDouble("interestRate");

int term = resultSet.getInt("loanTerm");

return (principal \* rate \* term) / 12;

} else {

throw new InvalidLoanException("Loan with ID " + loanId + " not found.");

}

} catch (SQLException e) {

e.printStackTrace();

return 0;

}

}

*@Override*

public double calculateInterest(double principal, double rate, int term) {

return (principal \* rate \* term) / 12;

}

*@Override*

public void loanStatus(int loanId) {

String fetchQuery = "select creditScore from Customer inner join Loan ON Customer.customerId = Loan.customerId where loanId = ?";

try (PreparedStatement statement = connection.prepareStatement(fetchQuery)) {

statement.setInt(1, loanId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

int creditScore = resultSet.getInt("creditScore");

String status = creditScore > 650 ? "Approved" : "Rejected";

String updateQuery = "UPDATE Loan SET loanStatus = ? where loanId = ?";

try (PreparedStatement updateStmt = connection.prepareStatement(updateQuery)) {

updateStmt.setString(1, status);

updateStmt.setInt(2, loanId);

updateStmt.executeUpdate();

System.***out***.println("Loan " + status);

}

} else {

System.***out***.println("Loan or customer not found.");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

*@Override*

public double calculateEMI(int loanId) throws InvalidLoanException {

String sql = "SELECT principalAmount, interestRate, loanTerm FROM Loan WHERE loanId = ?";

try (PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setInt(1, loanId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

double principal = resultSet.getDouble("principalAmount");

double rate = resultSet.getDouble("interestRate") / 12 / 100;

int term = resultSet.getInt("loanTerm");

return (principal \* rate \* Math.*pow*(1 + rate, term)) / (Math.*pow*(1 + rate, term) - 1);

} else {

throw new InvalidLoanException("Loan not found.");

}

} catch (SQLException e) {

e.printStackTrace();

return 0;

}

}

*@Override*

public double calculateEMI(double principal, double rate, int term) {

double monthlyRate = rate / 12 / 100;

return (principal \* monthlyRate \* Math.*pow*(1 + monthlyRate, term)) /

(Math.*pow*(1 + monthlyRate, term) - 1);

}

*@Override*

public void loanRepayment(int loanId, double amount) {

try {

double emi = calculateEMI(loanId);

if (amount < emi) {

System.***out***.println("Amount less than one EMI. Payment rejected.");

return;

}

int numberOfEmisPaid = (int) (amount / emi);

System.***out***.println("Number of EMIs paid: " + numberOfEmisPaid);

} catch (InvalidLoanException e) {

System.***out***.println(e.getMessage());

}

}

*@Override*

public List<Loan> getAllLoan() {

List<Loan> loanList = new ArrayList<>();

String sql = "SELECT \* FROM Loan";

try (Statement stmt = connection.createStatement();

ResultSet resultSet = stmt.executeQuery(sql)) {

while (resultSet.next()) {

Loan loan = new Loan();

loan.setLoanId(resultSet.getInt("loanId"));

loan.setPrincipalAmount(resultSet.getDouble("principalAmount"));

loan.setInterestRate(resultSet.getDouble("interestRate"));

loan.setLoanTerm(resultSet.getInt("loanTerm"));

loan.setLoanType(resultSet.getString("loanType"));

loan.setLoanStatus(resultSet.getString("loanStatus"));

loanList.add(loan);

}

} catch (SQLException e) {

e.printStackTrace();

}

return loanList;

}

*@Override*

public Loan getLoanById(int loanId) throws InvalidLoanException {

String sql = "SELECT \* FROM Loan WHERE loanId = ?";

try (PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setInt(1, loanId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

Loan loan = new Loan();

loan.setLoanId(resultSet.getInt("loanId"));

loan.setPrincipalAmount(resultSet.getDouble("principalAmount"));

loan.setInterestRate(resultSet.getDouble("interestRate"));

loan.setLoanTerm(resultSet.getInt("loanTerm"));

loan.setLoanType(resultSet.getString("loanType"));

loan.setLoanStatus(resultSet.getString("loanStatus"));

return loan;

} else {

throw new InvalidLoanException("Loan with ID " + loanId + " not found.");

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

}

Task 7: create database connection:   
  
package util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.util.Properties;

import java.io.FileInputStream;

public class DBUtil {

private static Connection *connection* = null;

public static Connection getDBConn() {

if (*connection* == null) {

try {

Properties props = new Properties();

FileInputStream fis = new FileInputStream("db.properties");

props.load(fis);

String url = props.getProperty("db.url");

String username = props.getProperty("db.username");

String password = props.getProperty("db.password");

if (url == null || username == null || password == null) {

throw new RuntimeException("One or more database properties (url, username, password) are missing in db.properties.");

}

*connection* = DriverManager.*getConnection*(url, username, password);

System.***out***.println("Database connection established.");

} catch (Exception e) {

System.***out***.println("Unexpected error while establishing DB connection.");

e.printStackTrace();

}

}

return *connection*;

}

}

Also creating a custom exception handling:  
  
package exception;

public class InvalidLoanException extends Exception {

public InvalidLoanException(String message) {

super(message);

}

}

Task 8: creating main method to inmplement menu such as "applyLoan", "getAllLoan", "getLoan",

"loanRepayment", "exit."  
  
package main;

import dao.LoanRepositoryImpl;

import dao.ILoanRepository;

import entity.CarLoan;

import entity.HomeLoan;

import entity.Customer;

import entity.Loan;

import exception.InvalidLoanException;

import java.util.\*;

public class LoanManagement {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

ILoanRepository loanRepo = new LoanRepositoryImpl();

boolean running = true;

while (running) {

System.out.println("\n--- Loan Management Menu ---");

System.out.println("1. Apply Loan");

System.out.println("2. Get All Loans");

System.out.println("3. Get Loan By ID");

System.out.println("4. Loan Repayment");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter loan ID: ");

int loanId = sc.nextInt();

System.out.print("Enter customer ID: ");

int customerId = sc.nextInt();

System.out.print("Enter principal amount: ");

double principal = sc.nextDouble();

System.out.print("Enter interest rate: ");

double rate = sc.nextDouble();

System.out.print("Enter loan term in months: ");

int term = sc.nextInt();

sc.nextLine(); // consume newline

System.out.print("Enter loan type (Home/Car): ");

String type = sc.nextLine().toLowerCase();

Customer cust = new Customer();

cust.setCustomerId(customerId);

Loan loan = null;

if (type.equals("home")) {

System.out.print("Enter property address: ");

String address = sc.nextLine();

System.out.print("Enter property value: ");

int value = sc.nextInt();

sc.nextLine();

loan = new HomeLoan(loanId, cust, principal, rate, term, "Home", "Pending", address, value);

} else if (type.equals("car")) {

System.out.print("Enter car model: ");

String model = sc.nextLine();

System.out.print("Enter car value: ");

int value = sc.nextInt();

sc.nextLine();

loan = new CarLoan(loanId, cust, principal, rate, term, "Car", "Pending", model, value);

} else {

System.out.println("Invalid loan type entered.");

}

if (loan != null) {

boolean success = loanRepo.applyLoan(loan);

System.out.println(success ? "Loan applied successfully!" : "Loan application failed.");

}

break;

case 2:

List<Loan> loans = loanRepo.getAllLoan();

for (Loan l : loans) {

System.out.println("Loan ID: " + l.getLoanId() + ", Type: " + l.getLoanType() + ", Status: " + l.getLoanStatus());

}

break;

case 3:

System.out.print("Enter loan ID: ");

int id = sc.nextInt();

try {

Loan l = loanRepo.getLoanById(id);

System.out.println("Loan Details:");

System.out.println("ID: " + l.getLoanId());

System.out.println("Type: " + l.getLoanType());

System.out.println("Status: " + l.getLoanStatus());

} catch (InvalidLoanException e) {

System.out.println("Error: " + e.getMessage());

}

break;

case 4:

System.out.print("Enter loan ID: ");

int repayId = sc.nextInt();

System.out.print("Enter repayment amount: ");

double amount = sc.nextDouble();

try {

loanRepo.loanRepayment(repayId, amount);

} catch (InvalidLoanException e) {

System.out.println("Repayment failed: " + e.getMessage());

}

break;

case 5:

running = false;

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Try again.");

}

}

sc.close();

}

}