Loan Management System

Name: Samiksha Sandeep Patil

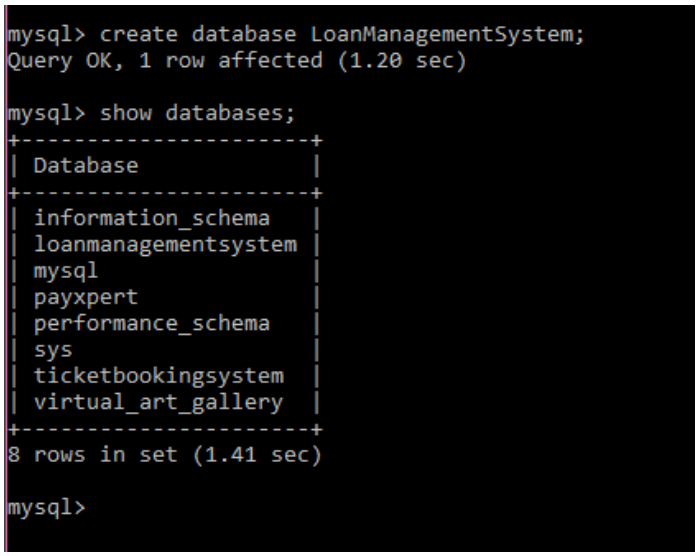
SuperSet Id: 5273555

Github Link: https://github.com/samikshapatil07/Loan_Management_System

Working with Database:

-- Create the Database

```
create database LoanManagementSystem;
show databases;
```



Created tables for:

1. `Customer`
2. `Loan` (base table for both HomeLoan and CarLoan)
3. `HomeLoan` (with foreign key to `Loan`)
4. `CarLoan` (with foreign key to `Loan`)

Create Customer Table

```
CREATE TABLE Customer (
customerId INT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
phone VARCHAR(15),
address VARCHAR(255),
creditScore INT
);
```

```
mysql> desc customer;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| customerId  | int          | NO   | PRI | NULL    |       |
| name        | varchar(100) | NO   |     | NULL    |       |
| email       | varchar(100) | NO   | UNI | NULL    |       |
| phone       | varchar(15)  | YES  |     | NULL    |       |
| address     | varchar(255) | YES  |     | NULL    |       |
| creditScore | int          | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.18 sec)
```

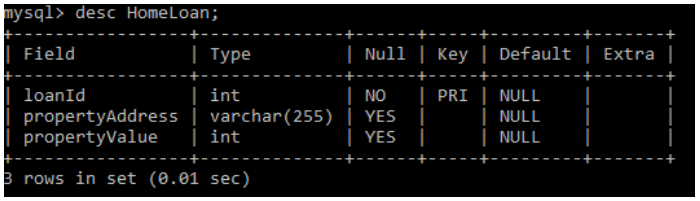
Create Loan Table

```
CREATE TABLE Loan (
loanId INT PRIMARY KEY,
customerId INT,
principalAmount DOUBLE,
interestRate DOUBLE,
loanTerm INT,
loanType VARCHAR(20),
loanStatus VARCHAR(20),
FOREIGN KEY (customerId) REFERENCES Customer(customerId)
);
```

```
mysql> desc loan;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| loanId          | int         | NO   | PRI | NULL    |       |
| customerId      | int         | YES  | MUL | NULL    |       |
| principalAmount | double      | YES  |     | NULL    |       |
| interestRate    | double      | YES  |     | NULL    |       |
| loanTerm        | int         | YES  |     | NULL    |       |
| loanType        | varchar(20) | YES  |     | NULL    |       |
| loanStatus      | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

Create HomeLoan Table

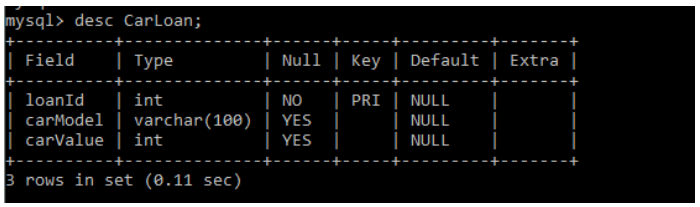
```
CREATE TABLE HomeLoan (
loanId INT PRIMARY KEY,
propertyAddress VARCHAR(255),
propertyValue INT,
FOREIGN KEY (loanId) REFERENCES Loan(loanId)
);
```



```
mysql> desc HomeLoan;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| loanId          | int          | NO   | PRI | NULL    |       |
| propertyAddress | varchar(255) | YES  |     | NULL    |       |
| propertyValue   | int          | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Create CarLoan Table

```
CREATE TABLE CarLoan (
loanId INT PRIMARY KEY,
carModel VARCHAR(100),
carValue INT,
FOREIGN KEY (loanId) REFERENCES Loan(loanId)
);
```



```
mysql> desc CarLoan;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| loanId   | int          | NO   | PRI | NULL    |       |
| carModel | varchar(100) | YES  |     | NULL    |       |
| carValue | int          | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.11 sec)
```

Inserted data into tables as shown below;

Customer table:

```
INSERT INTO Customer (customerId, name, email, phone, address, creditScor
(1, 'Samiksha', 'samiksha@example.com', '9876543210', '123 Ram nagar, Delh
(2, 'Sandeep', 'sandeep@example.com', '9123456780', '456 Kolhapur, MH', 64
(3, 'Madhu', 'madhu@example.com', '9988776655', '789 Pine Road, TN', 680)
```

```
mysql> select * from customer;
+------------+---------+---------------------+------------+---------------------+-------------+
| customerId | name    | email               | phone      | address             | creditScore |
+------------+---------+---------------------+------------+---------------------+-------------+
|          1 | Samiksha | samiksha@example.com | 9876543210 | 123 Ram nagar, Delhi |         720 |
|          2 | Sandeep  | sandeep@example.com  | 9123456780 | 456 Kolhapur, MH     |         640 |
|          3 | Madhu    | madhu@example.com    | 9988776655 | 789 Pine Road, TN    |         680 |
+------------+---------+---------------------+------------+---------------------+-------------+
3 rows in set (0.10 sec)
```

Loan table:

```
INSERT INTO Loan (loanId, customerId, principalAmount, interestRate, loanTer
 loanType, loanStatus) VALUES
(101, 1, 500000, 7.5, 60, 'HomeLoan', 'Pending'),
(102, 2, 300000, 8.0, 36, 'CarLoan', 'Pending'),
(103, 3, 600000, 6.8, 84, 'HomeLoan', 'Pending');
```

```
mysql> select * from Loan;
+--------+------------+-----------------+--------------+----------+----------+------------+
| loanId | customerId | principalAmount | interestRate | loanTerm | loanType | loanStatus |
+--------+------------+-----------------+--------------+----------+----------+------------+
|    101 |          1 |          500000 |          7.5 |       60 | HomeLoan | Pending    |
|    102 |          2 |          300000 |            8 |       36 | CarLoan  | Pending    |
|    103 |          3 |          600000 |          6.8 |       84 | HomeLoan | Pending    |
+--------+------------+-----------------+--------------+----------+----------+------------+
3 rows in set (0.00 sec)
```

HomeLoan table:

```
INSERT INTO HomeLoan (loanId, propertyAddress, propertyValue) VALUES
(101, '12 Shivaji Nagar, Delhi', 700000),
(103, '77 Anna Salai, TN', 800000);
```

```
mysql> select * from HomeLoan;
+--------+-------------------------+---------------+
| loanId | propertyAddress         | propertyValue |
+--------+-------------------------+---------------+
|    101 | 12 Shivaji Nagar, Delhi |        700000 |
|    103 | 77 Anna Salai, TN       |        800000 |
+--------+-------------------------+---------------+
2 rows in set (0.00 sec)
```

CarLone table:

```
INSERT INTO CarLoan (loanId, carModel, carValue) VALUES
(102, 'Hyundai Creta 2023', 350000);
```

```
mysql> select * from CarLoan;
+--------+--------------------+----------+
| loanId | carModel           | carValue |
+--------+--------------------+----------+
|    102 | Hyundai Creta 2023 |   350000 |
+--------+--------------------+----------+
1 row in set (0.00 sec)
```

Working with Java

Step 1: Create Packages

src/
│
├── entity → For data models (Customer, Loan, etc.)
├── dao → For interface and its implementation
├── exception → For custom exceptions
├── util → For DB utilities
└── main → For main class with menu

Step 2: Working in Entity Package

1. Create `Customer` Class:

- Customers : Stores basic customer info like name, phone, email, credit score.

```
package entity;
public class Customer {
    private int customerId;
    private String name;
    private String email;
    private String phone;
    private String address;
    private int creditScore;

    // Default constructor
    public Customer() {}
```

```
    // Parameterized constructor
    public Customer(int customerId, String name, String email, String phone,
    String address, int creditScore) {
        this.customerId = customerId;
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.address = address;
        this.creditScore = creditScore;
    }

    // Getters and setters
    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 getPhone() { return phone; }
    public void setPhone(String phone) { this.phone = phone; }

    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 ID: " + customerId + ", Name: " + name + ", Email: " + e
                ", Phone: " + phone + ", Address: " + address + ", Credit Score: " +
                creditScore;
    }
}
```

2. Create `Loan` Class:

- Loan : Base class with details like loan ID, amount, interest rate, and type.
- Acts as a **base class** for all types of loans.

```
package entity;

public class Loan {
    private int loanId;
    private Customer customer;
    private double principalAmount;
    private double interestRate;
    private int loanTerm; // in months
    private String loanType;
    private String loanStatus;

    // Default constructor
    public Loan() {}

    // Parameterized constructor
    public Loan(int loanId, Customer customer, double principalAmount, double
                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;
    }

    // Getters and setters
    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.principalAmo

    public double getInterestRate() { return interestRate; }
    public void setInterestRate(double interestRate) { this.interestRate = interest

    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;

    @Override
    public String toString() {
        return "Loan ID: " + loanId +
                ", Customer: [" + customer + "]" +
                ", Principal: " + principalAmount +
                ", Interest Rate: " + interestRate +
                ", Term: " + loanTerm + " months" +
                ", Type: " + loanType +
                ", Status: " + loanStatus;
    }
}
```

3. Create `HomeLoan` Class:

- Home : Special loan with property details (address, value).
- Extends `Loan`

```
package entity;

public class HomeLoan extends Loan {
    private String propertyAddress;
    private int propertyValue;
```

```
    // Default constructor
    public HomeLoan() {}

    // Parameterized constructor
    public HomeLoan(int loanId, Customer customer, double principalAmount, d
            String loanType, String loanStatus, String propertyAddress, int property
        super(loanId, customer, principalAmount, interestRate, loanTerm, loanType,
        this.propertyAddress = propertyAddress;
        this.propertyValue = propertyValue;
    }

    // Getters and setters
    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 String toString() {
        return super.toString() +
                ", Property Address: " + propertyAddress +
                ", Property Value: " + propertyValue;
    }
}
```

4. Create `CarLoan` Class:

- Car : Special loan with car model and value.
- Extends `Loan`

```
package entity;

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

    // Default constructor
    public CarLoan() {}

    // Parameterized constructor
    public CarLoan(int loanId, Customer customer, double principalAmount, dou
    interestRate,
                   int loanTerm, String loanType, String loanStatus,
                   String carModel, int carValue) {
        super(loanId, customer, principalAmount, interestRate, loanTerm, loanTyp
        loanStatus);
        this.carModel = carModel;
        this.carValue = carValue;
    }

    // Getters and setters
    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 String toString() {
        return super.toString() +
                ", Car Model: " + carModel +
                ", Car Value: " + carValue;
    }
}
```

Step 3: Working in dao Package

1. Create Interface `ILoanRepository`

- Declare methods like:

`applyLoan(), calculateInterest(), calculateEMI(), loanStatus(), loanRepayment(), getLoanById(), getAllLoan()`

```
package dao;

import entity.Loan;
import java.util.List;

public interface ILoanRepository {
    // Apply for a loan
    void applyLoan(Loan loan) throws Exception;

    //  Calculate interest using loanId
    double calculateInterest(int loanId) throws Exception;

    //  Overloaded: Calculate interest using parameters
    public default double calculateInterest(double principal, double rate, int terr
        return (principal * rate * term) / (12 * 100);
    }
    //  Update and return loan status based on credit score
    String loanStatus(int loanId) throws Exception;
```

```
    // Calculate EMI using loanId
    double calculateEMI(int loanId) throws Exception;

    // Overloaded: Calculate EMI using parameters
    double calculateEMI(double principal, double rate, int tenure);

    // Loan repayment
    void loanRepayment(int loanId, double amount) throws Exception;

    // Get all loans
    List<Loan> getAllLoan() throws Exception;

    // Get loan by ID
    Loan getLoanById(int loanId) throws Exception;
}
```

2. Implement in `ILoanRepositoryImpl`:

- Contains **all logic to interact with database** using JDBC.
- Separate logic based on `HomeLoan` or `CarLoan` using `instanceof`.
- Exception handling using `InvalidLoanException`.
- Calls SQL queries to insert, retrieve, update, or validate loan data.

```
package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import java.util.ArrayList;
import java.util.List;

import entity.Loan;
import entity.HomeLoan;
import entity.CarLoan;
```

```
import entity.Customer;

import exception.InvalidLoanException;

import util.DBConnUtil;

public class ILoanRepositoryImpl implements ILoanRepository {

    private Connection getConnection() throws Exception {
        return DBConnUtil.getDBConn(); // Utility method to get DB connection
    }

    /*-------------------------applyLoan---------------------------------*/
    @Override
    public void applyLoan(Loan loan) throws Exception {
        try (Connection conn = getConnection()) {

            // Insert into Loan table
            String loanSql = "INSERT INTO Loan (loanId, customerId, principalAmo
            interestRate, loanTerm, loanType, loanStatus) " +
                        "VALUES (?, ?, ?, ?, ?, ?, ?)";
            try (PreparedStatement ps = conn.prepareStatement(loanSql)) {
                ps.setInt(1, loan.getLoanId());
                ps.setInt(2, loan.getCustomer().getCustomerId());
                ps.setDouble(3, loan.getPrincipalAmount());
                ps.setDouble(4, loan.getInterestRate());
                ps.setInt(5, loan.getLoanTerm());
                ps.setString(6, loan.getLoanType());
                ps.setString(7, "Pending");
                ps.executeUpdate();
            }

            // Handle HomeLoan
            if (loan instanceof HomeLoan) {
                HomeLoan hl = (HomeLoan) loan;
                String sql = "INSERT INTO HomeLoan (loanId, propertyAddress,
                 propertyValue)VALUES (?, ?, ?)";
                try (PreparedStatement ps = conn.prepareStatement(sql)) {
```

```
                ps.setInt(1, hl.getLoanId());
                ps.setString(2, hl.getPropertyAddress());
                ps.setInt(3, hl.getPropertyValue());
                ps.executeUpdate();
            }

        // Handle CarLoan
        } else if (loan instanceof CarLoan) {
            CarLoan cl = (CarLoan) loan;
            String sql = "INSERT INTO CarLoan (loanId, carModel, carValue)
            VALUES (?, ?, ?)";
            try (PreparedStatement ps = conn.prepareStatement(sql)) {
                ps.setInt(1, cl.getLoanId());
                ps.setString(2, cl.getCarModel());
                ps.setInt(3, cl.getCarValue());
                ps.executeUpdate();
            }
        }

        System.out.println("Loan applied successfully and is in Pending status.
    }
}

/*-------------------------calculateInterest----------------------------*/
@Override
public double calculateInterest(int loanId) throws Exception {
    double interest = 0;

    try (Connection conn = getConnection()) {
        String query = "SELECT principalAmount, interestRate, loanTerm
        FROM Loan WHERE loanId = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, loanId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                throw new InvalidLoanException("Loan not found for ID: " + loanId
            }
```

```
            double principal = rs.getDouble("principalAmount");
            double rate = rs.getDouble("interestRate");
            int term = rs.getInt("loanTerm");

            interest = (principal * rate * term) / (12 * 100);
        }
    }

    System.out.println("Calculated Interest for Loan ID " + loanId + ": " + inter
    return interest;
}
  /*-------------------------loanStatus-----------------------------------
@Override
public String loanStatus(int loanId) throws Exception {
    String status;

    try (Connection conn = getConnection()) {
        String query = "SELECT L.loanId, C.creditScore FROM Loan L " +
                       "JOIN Customer C ON L.customerId = C.customerId " +
                       "WHERE L.loanId = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, loanId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                throw new InvalidLoanException("Loan ID not found: " + loanId);
            }

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

            if (creditScore > 650) {
                status = "Approved";
            } else {
                status = "Rejected";
            }

            // Update loan status in DB
```

```
            String updateQuery = "UPDATE Loan SET loanStatus = ? WHERE loa
            try (PreparedStatement updatePs = conn.prepareStatement(updateQ
                updatePs.setString(1, status);
                updatePs.setInt(2, loanId);
                updatePs.executeUpdate();
            }
        }
    }

    System.out.println("Loan ID " + loanId + " is now " + status);
    return status;
}
```

```
/*-------------------------calculateEMI-----------------------------------
@Override
public double calculateEMI(int loanId) throws Exception {
    double emi;

    try (Connection conn = getConnection()) {
        String query = "SELECT principalAmount, interestRate, loanTerm FROM
        Loan WHERE loanId = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, loanId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                throw new InvalidLoanException("Loan not found for ID: " + loanId
            }

            double principal = rs.getDouble("principalAmount");
            double rate = rs.getDouble("interestRate") / 12 / 100;
            int tenure = rs.getInt("loanTerm");

            emi = (principal * rate * Math.pow(1 + rate, tenure)) /
                  (Math.pow(1 + rate, tenure) - 1);
        }
```

```
    }

    System.out.println("EMI for Loan ID " + loanId + ": " + emi);
    return emi;
}

@Override
public double calculateEMI(double principal, double rate, int tenure) {
    double monthlyRate = rate / 12 / 100;
    return (principal * monthlyRate * Math.pow(1 + monthlyRate, tenure)) /
           (Math.pow(1 + monthlyRate, tenure) - 1);
}

/*-------------------------loanRepayment---------------------------------
@Override
public void loanRepayment(int loanId, double amount) throws Exception {
    try (Connection conn = getConnection()) {
        String query = "SELECT principalAmount, interestRate, loanTerm FROM
        Loan WHERE loanId = ?";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setInt(1, loanId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                throw new InvalidLoanException("Loan not found for ID: " + loanId
            }

            double principal = rs.getDouble("principalAmount");
            double rate = rs.getDouble("interestRate");
            int tenure = rs.getInt("loanTerm");

            double emi = calculateEMI(principal, rate, tenure);

            if (amount < emi) {
                System.out.println("Repayment failed: Amount is less than
                single EMI (" + emi + ")");
            } else {
                int emiCount = (int) (amount / emi);
```

```
            System.out.println("Repayment successful. You have paid " +
            emiCount + " EMI(s).");
        }
      }
   }
}

/*-------------------------getAllLoan-------------------------------------
@Override
public List<Loan> getAllLoan() throws Exception {
   List<Loan> loanList = new ArrayList<>();

   try (Connection conn = getConnection()) {
      String loanQuery = "SELECT * FROM Loan";
      try (PreparedStatement ps = conn.prepareStatement(loanQuery)) {
         ResultSet rs = ps.executeQuery();

         while (rs.next()) {
            int loanId = rs.getInt("loanId");
            int customerId = rs.getInt("customerId");
            double principal = rs.getDouble("principalAmount");
            double rate = rs.getDouble("interestRate");
            int term = rs.getInt("loanTerm");
            String type = rs.getString("loanType");
            String status = rs.getString("loanStatus");

            Customer customer = getCustomerById(customerId, conn);
            Loan loan;

            if ("HomeLoan".equalsIgnoreCase(type)) {
               String homeQuery = "SELECT * FROM HomeLoan WHERE loanI
               try (PreparedStatement ps2 = conn.prepareStatement(homeQu
                  ps2.setInt(1, loanId);
                  ResultSet rs2 = ps2.executeQuery();
                  if (rs2.next()) {
                     String address = rs2.getString("propertyAddress");
                     int value = rs2.getInt("propertyValue");
                     loan = new HomeLoan(loanId, customer, principal, rate,
```

```
                    term, type, status, address, value);
                } else continue;
            }
        } else if ("CarLoan".equalsIgnoreCase(type)) {
            String carQuery = "SELECT * FROM CarLoan WHERE loanId = ?
            try (PreparedStatement ps2 = conn.prepareStatement(carQuery
                ps2.setInt(1, loanId);
                ResultSet rs2 = ps2.executeQuery();
                if (rs2.next()) {
                    String model = rs2.getString("carModel");
                    int value = rs2.getInt("carValue");
                    loan = new CarLoan(loanId, customer, principal, rate,
                    term, type, status, model, value);
                } else continue;
            }
        } else {
            loan = new Loan(loanId, customer, principal, rate, term,
            type, status);
        }

        loanList.add(loan);
      }
    }
  }

  for (Loan loan : loanList) {
    System.out.println(loan);
  }

  return loanList;
}

/*-------------------------getLoanById-----------------------------------*
@Override
public Loan getLoanById(int loanId) throws Exception {
  Loan loan = null;

  try (Connection conn = getConnection()) {
```

```
        String loanQuery = "SELECT * FROM Loan WHERE loanId = ?";
        try (PreparedStatement ps = conn.prepareStatement(loanQuery)) {
            ps.setInt(1, loanId);
            ResultSet rs = ps.executeQuery();

            if (!rs.next()) {
                throw new InvalidLoanException("Loan not found for ID: " + loanId
            }

            int customerId = rs.getInt("customerId");
            double principal = rs.getDouble("principalAmount");
            double rate = rs.getDouble("interestRate");
            int term = rs.getInt("loanTerm");
            String type = rs.getString("loanType");
            String status = rs.getString("loanStatus");

            Customer customer = getCustomerById(customerId, conn);

            if ("HomeLoan".equalsIgnoreCase(type)) {
                String homeQuery = "SELECT * FROM HomeLoan WHERE loanId =
                try (PreparedStatement ps2 = conn.prepareStatement(homeQuery
                    ps2.setInt(1, loanId);
                    ResultSet rs2 = ps2.executeQuery();
                    if (rs2.next()) {
                        String address = rs2.getString("propertyAddress");
                        int value = rs2.getInt("propertyValue");
                        loan = new HomeLoan(loanId, customer, principal, rate,
                        term, type, status, address, value);
                    }
                }
            } else if ("CarLoan".equalsIgnoreCase(type)) {
                String carQuery = "SELECT * FROM CarLoan WHERE loanId = ?";
                try (PreparedStatement ps2 = conn.prepareStatement(carQuery))
                    ps2.setInt(1, loanId);
                    ResultSet rs2 = ps2.executeQuery();
                    if (rs2.next()) {
                        String model = rs2.getString("carModel");
                        int value = rs2.getInt("carValue");
```

```
                    loan = new CarLoan(loanId, customer, principal, rate,
                    term, type, status, model, value);
                }
            }
        } else {
            loan = new Loan(loanId, customer, principal, rate, term, type, statu
        }
    }

    System.out.println(loan);
    return loan;
}

private Customer getCustomerById(int customerId, Connection conn) throw
    String query = "SELECT * FROM Customer WHERE customerId = ?";
    try (PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setInt(1, customerId);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return new Customer(
                rs.getInt("customerId"),
                rs.getString("name"),
                rs.getString("email"),
                rs.getString("phone"),
                rs.getString("address"),
                rs.getInt("creditScore")
            );
        }
    }
    return null;
}
}
```

Step 4: Working in `exception` Package

1. Create `InvalidLoanException` Class:

- Extends `Exception`
- Used when a loan ID is not found
- Helps handle errors more cleanly in the app.

```
package exception;

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

Step 5: Working in `util` Package

1. Create `DBPropertyUtil`:

- Reads database configuration from `db.properties` file

```
package util;

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

public class DBPropertyUtil {

    public static String getConnectionString(String fileName) {
        Properties props = new Properties();
        String connStr = null;

        try (FileInputStream fis = new FileInputStream(fileName)) {
            props.load(fis);
            String url = props.getProperty("db.url");
            String user = props.getProperty("db.user");
            String password = props.getProperty("db.password");

            connStr = url + "?user=" + user + "&password=" + password;
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }

        return connStr;
    }
}
```

2. Create `DBConnUtil`:

- Uses `DBPropertyUtil` to create and return a `Connection` object

```
package util;

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

public class DBConnUtil {

    public static Connection getDBConn() throws Exception {
        // Update this with your actual path if needed
        String fileName = "db.properties";
        String connStr = DBPropertyUtil.getConnectionString(fileName);

        return DriverManager.getConnection(connStr);
    }
}
```

1. Create db.properties:

```
db.url=jdbc:mysql://localhost:3306/loanmanagementsystem
db.user=root
db.password=Spatil@07
```

Step 7: Create `LoanManagement.java`

1. Apply Loan

2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit

```
package main;

import dao.ILoanRepository;
import dao.ILoanRepositoryImpl;
import entity.*;

import java.util.Scanner;

public class LoanManagement {

    public static void main(String[] args) {
        ILoanRepository repo = new ILoanRepositoryImpl();
        Scanner scanner = new Scanner(System.in);
        int choice = 0;

        System.out.println("=== Welcome to Loan Management System ===");

        while (choice != 7) {
            System.out.println("\n==== MENU ====");
            System.out.println("1. Apply Loan");
            System.out.println("2. View All Loans");
            System.out.println("3. Get Loan by ID");
            System.out.println("4. Calculate Interest");
            System.out.println("5. Check Loan Status");
            System.out.println("6. Repay Loan");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
```

```
        switch (choice) {
            case 1:
                System.out.println("Enter Loan Type (H for HomeLoan, C for CarL
                String type = scanner.next();

                System.out.println("Loan ID: ");
                int loanId = scanner.nextInt();

                System.out.println("Customer ID: ");
                int customerId = scanner.nextInt();
                Customer customer = new Customer(customerId, "", "", "", "", 0);

                System.out.println("Principal Amount: ");
                double principal = scanner.nextDouble();

                System.out.println("Interest Rate: ");
                double rate = scanner.nextDouble();

                System.out.println("Loan Term (months): ");
                int term = scanner.nextInt();
                scanner.nextLine(); // clear buffer

                if (type.equals("H")) {
                    System.out.println("Property Address: ");
                    String address = scanner.nextLine();
                    System.out.println("Property Value: ");
                    int value = scanner.nextInt();

                    HomeLoan HomeLoan = new HomeLoan(loanId, customer, princ
                    try {
                        repo.applyLoan(HomeLoan);
                    } catch (Exception e) {
                        System.out.println("Error: " + e.getMessage());
                    }

                } else if (type.equals("C")) {
                    System.out.println("Car Model: ");
                    String model = scanner.nextLine();
```

```
            System.out.println("Car Value: ");
            int value = scanner.nextInt();

            CarLoan carLoan = new CarLoan(loanId, customer, principal, ra
            try {
                repo.applyLoan(carLoan);
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }

        } else {
            System.out.println("Invalid loan type.");
        }
        break;

    case 2:
        try {
            repo.getAllLoan();
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
        break;

    case 3:
        System.out.println("Enter Loan ID: ");
        int id = scanner.nextInt();
        try {
            repo.getLoanById(id);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
        break;

    case 4:
        System.out.println("Enter Loan ID: ");
        int interestId = scanner.nextInt();
        try {
            repo.calculateInterest(interestId);
```

```
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 5:
            System.out.println("Enter Loan ID: ");
            int statusId = scanner.nextInt();
            try {
                repo.loanStatus(statusId);
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 6:
            System.out.println("Enter Loan ID: ");
            int repayId = scanner.nextInt();
            System.out.println("Enter Repayment Amount: ");
            double repayAmount = scanner.nextDouble();
            try {
                repo.loanRepayment(repayId, repayAmount);
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;

        case 7:
            System.out.println("Exiting Loan Management System. Goodbye!"
            break;

        default:
            System.out.println("Invalid option. Try again.");
    }
}

scanner.close();
```

```
    }
  }
```

Output:

1. Apply Loan:

a. Home Loan:

```
=== Welcome to Loan Management System ===

==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 1
Enter Loan Type (H for HomeLoan, C for CarLoan):
H
Loan ID:
106
Customer ID:
5
Principal Amount:
3000000
Interest Rate:
10
Loan Term (months):
100
Property Address:
Property Road,Delhi
Property Value:
5000000
Loan applied successfully and is in Pending status.
```

b. CarLoan:

```
==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 1
Enter Loan Type (H for HomeLoan, C for CarLoan):
C
Loan ID:
107
Customer ID:
5
Principal Amount:
100000
Interest Rate:
9
Loan Term (months):
80
Car Model:
Verna
Car Value:
100000
Loan applied successfully and is in Pending status.
```

2.View All Loans

```
==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 2
Loan ID: 101, Customer: [Customer ID: 1, Name: Samiksha, Email: samiksha@example.com, Phone: 9876543210, Address: 123 Ram nagar, Delhi, Credit Score: 720], Principal: 500
Loan ID: 102, Customer: [Customer ID: 2, Name: Sandeep, Email: sandeep@example.com, Phone: 9123456780, Address: 456 Kolhapur, MH, Credit Score: 640], Principal: 300000.0,
Loan ID: 103, Customer: [Customer ID: 3, Name: Madhu, Email: madhu@example.com, Phone: 9988776655, Address: 789 Pine Road, TN, Credit Score: 680], Principal: 600000.0, In
Loan ID: 104, Customer: [Customer ID: 4, Name: Amit, Email: amit@example.com, Phone: 9876540000, Address: 101 MG Road, Pune, Credit Score: 700], Principal: 1000000.0, Int
Loan ID: 105, Customer: [Customer ID: 4, Name: Amit, Email: amit@example.com, Phone: 9876540000, Address: 101 MG Road, Pune, Credit Score: 700], Principal: 1000000.0, Int
Loan ID: 106, Customer: [Customer ID: 5, Name: Anuj, Email: anuj@example.com, Phone: 9876540000, Address: 18 Gandhi Road, Banglore, Credit Score: 800], Principal: 3000000
Loan ID: 107, Customer: [Customer ID: 5, Name: Anuj, Email: anuj@example.com, Phone: 9876540000, Address: 18 Gandhi Road, Banglore, Credit Score: 800], Principal: 100000.
```

3.Get Loan by ID

```
=== Welcome to Loan Management System ===

==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 3
Enter Loan ID:
107
Loan ID: 107, Customer: [Customer ID: 5, Name: Anuj, Email: anuj@example.com, Phone: 9876540000, Address: 18 Gandhi Road, Banglore, Credit Score:
```

4.Calculate Interest

```
=== Welcome to Loan Management System ===

==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 4
Enter Loan ID:
107
Calculated Interest for Loan ID 107: 60000.0
```

5.Check Loan Status

```
==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 5
Enter Loan ID:
107
Loan ID 107 is now Approved
```

6.Repay Loan

```
==== MENU ====
1. Apply Loan
2. View All Loans
3. Get Loan by ID
4. Calculate Interest
5. Check Loan Status
6. Repay Loan
7. Exit
Enter your choice: 6
Enter Loan ID:
107
Enter Repayment Amount:
100000
Repayment successful. You have paid 59 EMI(s).
```