Java Coding Challenge Insurance Management System

Instructions

- Submitting assignments should be a single file.
- Document should contain source code with output screenshot, corresponding to each question along with project structure screenshot.
- Single uploaded document should contain source code for all the questions.
- Each assignment builds upon the previous one, and by the end, you will have a comprehensive **Insurance** Management System implemented in Java with a strong focus on SQL schema design, control flow statements, loops, arrays, collections, and database interaction using JDBC.
- Remember to provide clear instructions, sample code, and any necessary resources or databases for each assignment.
- Follow object-oriented principles throughout the Java programming assignments. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exception from method and handle in the main method**.**

1: Problem Statement:

Insurance Management System using Core Java with JDBC

Introduction: The objective of this project is to create an Insurance Management System using Core Java and JDBC for a fictitious insurance company. The system should allow the company to manage policies, clients, claims, and premium payments. The project aims to improve the efficiency and organization of the insurance processes and ensure accurate record-keeping**.**

1. Schema design:

- The Users table manages user information, including their roles.
- The Policies table stores details about insurance policies, including policy type, coverage, and premium amounts.
- The Clients table is used to manage client information, linked to their associated policies.
- The Claims table records insurance claims with their statuses, linked to the associated policies and clients.
- The Payments table records premium payments made by clients, associated with client records.

```
-- Create the Users table to manage system users with different roles
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,
    role ENUM('admin', 'agent', 'client') NOT NULL
);
-- Create the Policies table to store information about insurance policies
CREATE TABLE Policies (
    policy_id INT PRIMARY KEY AUTO_INCREMENT,
    policy_number VARCHAR(20) NOT NULL,
    policy_type VARCHAR(50) NOT NULL,
    coverage_amount DECIMAL(10, 2) NOT NULL,
```

```
    premium_amount DECIMAL(8, 2) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL
);
-- Create the Clients table to manage client information
CREATE TABLE Clients (
    client_id INT PRIMARY KEY AUTO_INCREMENT,
    client_name VARCHAR(100) NOT NULL,
    contact_info VARCHAR(255),
    policy_id INT,
    FOREIGN KEY (policy_id) REFERENCES Policies(policy_id));
-- Create the Claims table to record insurance claims
CREATE TABLE Claims (
    claim_id INT PRIMARY KEY AUTO_INCREMENT,
    claim_number VARCHAR(20) NOT NULL,
    date_filed DATE NOT NULL,
    claim_amount DECIMAL(10, 2) NOT NULL,
    status ENUM('pending', 'approved', 'denied') NOT NULL,
    policy_id INT,
    client_id INT,
    FOREIGN KEY (policy_id) REFERENCES Policies(policy_id),
    FOREIGN KEY (client_id) REFERENCES Clients(client_id));
-- Create the Payments table to record premium payments
```

```
CREATE TABLE Payments (
    payment_id INT PRIMARY KEY AUTO_INCREMENT,
    payment_date DATE NOT NULL,
    payment_amount DECIMAL(8, 2) NOT NULL,
    client_id INT,
    FOREIGN KEY (client_id) REFERENCES Clients(client_id));
```

2: Java Model classes

5: Object Oriented Programming

Scope : Entity classes/Models/POJO, Abstraction/Encapsulation

Create the following **model/entity classes** within package **com.hexaware.entities** with variables declared private, constructors(default and parametrized,getters,setters and toString())

1. **User Class:**

```
public class User {
    private int userId;
    private String username;
    private String password;
    private UserRole role;
    // Constructors, getters, and setters
}
public enum UserRole {
    ADMIN, AGENT, CLIENT
}
```

Client Class:

```
public class Client {
    private int clientId;
    private String clientName;
    private String contactInfo;
 private Policy policy;//Represents the policy associated with the client
    // Constructors, getters, and setters}
```

Claim Class

```
import java.util.Date;
public class Claim {
    private int claimId;
    private String claimNumber;
    private Date dateFiled;
    private double claimAmount;
    private ClaimStatus status;
private Policy policy;//Represents the policy associated with the claim
private Client client; // Represents the client associated with the claim

    // Constructors, getters, and setters
}
public enum ClaimStatus {
    PENDING, APPROVED, DENIED
```

}

Payment Class:

```
import java.util.Date;
public class Payment {
    private int paymentId;
    private Date paymentDate;
    private double paymentAmount;
    private Client client;
   // Represents the client associated with the payment
    // Constructors, getters, and setters
}
```

6: Service Provider Interface

Keep the interfaces and implementation classes in package com.hexaware.dao

Create IPolicyService Interface

This interface can define methods for managing insurance policies.

```
public interface IPolicyService {
    boolean createPolicy(Policy policy);
    Policy getPolicy(int policyId);
    List<Policy> getAllPolicies();
    boolean updatePolicy(Policy policy);
    boolean deletePolicy(int policyId);
}
```

7: JDBC and Database Interaction

Scope: Connection class, Resultset, execute(),executeQuery(),SQL exception, datatype compatibility

Task: Connect your Java application to the SQL database using JDBC**:**

1. Write Java code to establish a connection to your SQL database.

   Create a utility class **DBConnection** in a package **com.hexaware.util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

   Connection properties supplied in the connection string should be read from a property file.

   Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

7: Service implementation

(Scope: Interfaces, Implementation classes, Object Arrays, Collections)

1. Create a Service class **IPolicyServiceImpl** in **com.hexaware.dao** with a static variable named connection of type **Connection** which can be assigned in the constructor by invoking the **getConnection()** method in **DBConnection** class

2. Provide implementation for all the methods in the interface by using jdbc.

8: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,thow & throws keyword usage)

Create the exceptions in package com.hexaware.myexceptions

Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

1. **PolicyNumberNotFoundException** :throw this exception when user enters an invalid policy number which doesn't exist in db

9. Main Method

Create class named MainModule with main method in com.hexware.mainmod.

Trigger all the methods in service implementation class.