Coding Challenge: Insurance Management System

- Project submissions should be done through the partcipants' Github repository and the link should be shared with trainers and Hexavarsity.
- 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**
        - 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:

1**Create SQL Schema from the following classes class, use the class attributes for table column names.**

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

1. Define **`User`** class with the following confidential attributes:

    a. userId;
    b. username;
    c. password;
    d. role;

© Hexaware Technologies Limited. All rights

2. Define ` **Client** ` class with the following confidential attributes:
    a. clientId;
    b. clientName;
    c. contactInfo;
    d. policy;//Represents the policy associated with the client

3. Define ` Claim ` class with the following confidential attributes:
    a. claimId;
    b. claimNumber;
    c. dateFiled;
    d. claimAmount;
    e. status;
    f. policy;//Represents the policy associated with the claim
    g. client; // Represents the client associated with the claim

4.. Define ` Claim ` class with the following confidential attributes:

    a. paymentId;
    b. paymentDate;
    c. paymentAmount;
    d. client; // Represents the client associated with the payment

2. Implement the following for all model classes. Write default constructors and overload the constructor with parameters, getters and setters, method to print all the member variables and values.

3. Define IPolicyService interface/abstract class with following methods to interact with database Keep the interfaces and implementation classes in package dao

a. createPolicy()

    I. parameters: Policy Object
    II. return type: boolean

b. getPolicy()

    I. parameters: policyId
    II. return type: Policy Object

c.getAllPolicies()

    I. parameters: none
    II. return type: Collection of Policy Objects

d.updatePolicy()

    I. parameters: Policy Object
    II. return type: boolean

© Hexaware Technologies Limited. All rights

e. deletePolicy()

I. parameters: PolicyId
II. return type: boolean

6. Define **InsuranceServiceImpl** class and implement all the methods **InsuranceServiceImpl .**

7. Create a utility class **DBConnection** in a package **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.

8. Create the exceptions in package myexceptions

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

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

9. Create class named MainModule with main method in package mainmod.

Trigger all the methods in service implementation class.

© Hexaware Technologies Limited. All rights