**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.**

**o entity**

**▪ Create entity classes in this package. All entity class should not have any**

**business logic.**

**o dao**

**▪ Create Service Provider interface to showcase functionalities.**

**▪ Create the implementation class for the above interface with db interaction.**

**o exception**

**▪ Create user defined exceptions in this package and handle exceptions whenever**

**needed.**

**o 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).**

**o main**

**▪ Create a class MainModule and demonstrate the functionalities in a menu**

**driven application.**

**Problem Statement:**

**1Create 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())**

**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.**

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

**a. userId;**

**b. username;**

**c. password;**

**d. role;**

|  |
| --- |
| **package** entity;  **public** **class** User {  **private** **int** userId;  **private** String username;  **private** String password;  **private** String role;  **public** User() {}  **public** User(**int** userId, String username, String password, String role) {  **this**.userId = userId;  **this**.username = username;  **this**.password = password;  **this**.role = role;  }  **public** **int** getUserId() {  **return** userId;  }  **public** **void** setUserId(**int** userId) {  **this**.userId = userId;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** String getRole() {  **return** role;  }  **public** **void** setRole(String role) {  **this**.role = role;  }  @Override  **public** String toString() {  **return** "User [userId=" + userId + ", username=" + username + ", role=" + role + "]";  }  } |

**2. Define ` Client ` class with the following confidential attributes:**

**a. clientId;**

**b. clientName;**

**c. contactInfo;**

**d. policy;//Represents the policy associated with the client**

|  |
| --- |
| **package** entity;  **public** **class** Client {  **private** **int** clientId;  **private** String clientName;  **private** String contactInfo;  **private** **int** policyId;  **public** Client() {}  **public** Client(**int** clientId, String clientName, String contactInfo, **int** policyId) {  **this**.clientId = clientId;  **this**.clientName = clientName;  **this**.contactInfo = contactInfo;  **this**.policyId = policyId;  }  **public** **int** getClientId() {  **return** clientId;  }  **public** **void** setClientId(**int** clientId) {  **this**.clientId = clientId;  }  **public** String getClientName() {  **return** clientName;  }  **public** **void** setClientName(String clientName) {  **this**.clientName = clientName;  }  **public** String getContactInfo() {  **return** contactInfo;  }  **public** **void** setContactInfo(String contactInfo) {  **this**.contactInfo = contactInfo;  }  **public** **int** getPolicyId() {  **return** policyId;  }  **public** **void** setPolicyId(**int** policyId) {  **this**.policyId = policyId;  }  @Override  **public** String toString() {  **return** "Client [clientId=" + clientId + ", clientName=" + clientName +  ", contactInfo=" + contactInfo + ", policyId=" + policyId + "]";  }  } |

**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**

|  |
| --- |
| **package** entity;  **import** java.sql.Date;  **public** **class** Claim {  **private** **int** claimId;  **private** String claimNumber;  **private** Date dateFiled;  **private** **double** claimAmount;  **private** String status;  **private** **int** policyId;  **private** **int** clientId;  **public** Claim() {}  **public** Claim(**int** claimId, String claimNumber, Date dateFiled, **double** claimAmount, String status, **int** policyId, **int** clientId) {  **this**.claimId = claimId;  **this**.claimNumber = claimNumber;  **this**.dateFiled = dateFiled;  **this**.claimAmount = claimAmount;  **this**.status = status;  **this**.policyId = policyId;  **this**.clientId = clientId;  }  **public** **int** getClaimId() {  **return** claimId;  }  **public** **void** setClaimId(**int** claimId) {  **this**.claimId = claimId;  }  **public** String getClaimNumber() {  **return** claimNumber;  }  **public** **void** setClaimNumber(String claimNumber) {  **this**.claimNumber = claimNumber;  }  **public** Date getDateFiled() {  **return** dateFiled;  }  **public** **void** setDateFiled(Date dateFiled) {  **this**.dateFiled = dateFiled;  }  **public** **double** getClaimAmount() {  **return** claimAmount;  }  **public** **void** setClaimAmount(**double** claimAmount) {  **this**.claimAmount = claimAmount;  }  **public** String getStatus() {  **return** status;  }  **public** **void** setStatus(String status) {  **this**.status = status;  }  **public** **int** getPolicyId() {  **return** policyId;  }  **public** **void** setPolicyId(**int** policyId) {  **this**.policyId = policyId;  }  **public** **int** getClientId() {  **return** clientId;  }  **public** **void** setClientId(**int** clientId) {  **this**.clientId = clientId;  }  @Override  **public** String toString() {  **return** "Claim [claimId=" + claimId + ", claimNumber=" + claimNumber + ", dateFiled=" + dateFiled +  ", claimAmount=" + claimAmount + ", status=" + status + ", policyId=" + policyId +  ", clientId=" + clientId + "]";  }  } |

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

**a. paymentId;**

**b. paymentDate;**

**c. paymentAmount;**

**d. client; // Represents the client associated with the payment**

|  |
| --- |
| **package** entity;  **import** java.sql.Date;  **public** **class** Payment {  **private** **int** paymentId;  **private** Date paymentDate;  **private** **double** paymentAmount;  **private** **int** clientId;  **public** Payment() {}  **public** Payment(**int** paymentId, Date paymentDate, **double** paymentAmount, **int** clientId) {  **this**.paymentId = paymentId;  **this**.paymentDate = paymentDate;  **this**.paymentAmount = paymentAmount;  **this**.clientId = clientId;  }  **public** **int** getPaymentId() {  **return** paymentId;  }  **public** **void** setPaymentId(**int** paymentId) {  **this**.paymentId = paymentId;  }  **public** Date getPaymentDate() {  **return** paymentDate;  }  **public** **void** setPaymentDate(Date paymentDate) {  **this**.paymentDate = paymentDate;  }  **public** **double** getPaymentAmount() {  **return** paymentAmount;  }  **public** **void** setPaymentAmount(**double** paymentAmount) {  **this**.paymentAmount = paymentAmount;  }  **public** **int** getClientId() {  **return** clientId;  }  **public** **void** setClientId(**int** clientId) {  **this**.clientId = clientId;  }  @Override  **public** String toString() {  **return** "Payment [paymentId=" + paymentId + ", paymentDate=" + paymentDate +  ", paymentAmount=" + paymentAmount + ", clientId=" + clientId + "]";  }  } |

**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**

**e. deletePolicy()**

**I. parameters: PolicyId**

**II. return type: boolean**

|  |
| --- |
| package dao;  import entity.Policy;  import util.DBConnection;  import java.sql.\*;  import java.util.\*;  public class PolicyDAOImpl implements IPolicyDAO {  private static final Connection connection = DBConnection.getConnection();  @Override  public boolean addPolicy(Policy policy) {  String sql = "INSERT INTO policies (policy\_name, coverage\_amount, premium) VALUES (?, ?, ?)";  try (PreparedStatement ps = connection.prepareStatement(sql)) {  ps.setString(1, policy.getPolicyName());  ps.setDouble(2, policy.getCoverageAmount());  ps.setDouble(3, policy.getPremium());  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Error in addPolicy: " + e.getMessage());  return false;  }  }  @Override  public boolean updatePolicy(Policy policy) {  String sql = "UPDATE policies SET policy\_name = ?, coverage\_amount = ?, premium = ? WHERE policy\_id = ?";  try (PreparedStatement ps = connection.prepareStatement(sql)) {  ps.setString(1, policy.getPolicyName());  ps.setDouble(2, policy.getCoverageAmount());  ps.setDouble(3, policy.getPremium());  ps.setInt(4, policy.getPolicyId());  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Error in updatePolicy: " + e.getMessage());  return false;  }  }  @Override  public boolean deletePolicy(int policyId) {  String sql = "DELETE FROM policies WHERE policy\_id = ?";  try (PreparedStatement ps = connection.prepareStatement(sql)) {  ps.setInt(1, policyId);  return ps.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println("Error in deletePolicy: " + e.getMessage());  return false;  }  }  @Override  public Policy getPolicyById(int policyId) {  String sql = "SELECT \* FROM policies WHERE policy\_id = ?";  try (PreparedStatement ps = connection.prepareStatement(sql)) {  ps.setInt(1, policyId);  ResultSet rs = ps.executeQuery();  if (rs.next()) {  return new Policy(  rs.getInt("policy\_id"),  rs.getString("policy\_name"),  rs.getDouble("coverage\_amount"),  rs.getDouble("premium"));  }  } catch (SQLException e) {  System.out.println("Error in getPolicyById: " + e.getMessage());  }  return null;  }  @Override  public List<Policy> getAllPolicies() {  List<Policy> policies = new ArrayList<>();  String sql = "SELECT \* FROM policies";  try (Statement stmt = connection.createStatement();  ResultSet rs = stmt.executeQuery(sql)) {  while (rs.next()) {  policies.add(new Policy(  rs.getInt("policy\_id"),  rs.getString("policy\_name"),  rs.getDouble("coverage\_amount"),  rs.getDouble("premium")));  }  } catch (SQLException e) {  System.out.println("Error in getAllPolicies: " + e.getMessage());  }  return policies;  }  } |

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

|  |
| --- |
| package dao;  import dao.\*;  import entity.\*;  import java.util.List;  public class InsuranceServiceImpl {  private final IUserDAO userDAO = new UserDAOImpl();  private final IPolicyDAO policyDAO = new PolicyDAOImpl();  private final IClientDAO clientDAO = new ClientDAOImpl();  private final IClaimDAO claimDAO = new ClaimDAOImpl();  private final IPaymentDAO paymentDAO = new PaymentDAOImpl();    public boolean registerUser(User user) {  return userDAO.registerUser(user);  }  public User loginUser(String username, String password) {  return userDAO.loginUser(username, password);  }    public boolean addPolicy(Policy policy) {  return policyDAO.addPolicy(policy);  }  public boolean updatePolicy(Policy policy) {  return policyDAO.updatePolicy(policy);  }  public boolean deletePolicy(int policyId) {  return policyDAO.deletePolicy(policyId);  }  public Policy getPolicyById(int policyId) {  return policyDAO.getPolicyById(policyId);  }  public List<Policy> getAllPolicies() {  return policyDAO.getAllPolicies();  }    public boolean addClient(Client client) {  return clientDAO.addClient(client);  }  public boolean updateClient(Client client) {  return clientDAO.updateClient(client);  }  public boolean deleteClient(int clientId) {  return clientDAO.deleteClient(clientId);  }  public Client getClientById(int clientId) {  return clientDAO.getClientById(clientId);  }  public List<Client> getAllClients() {  return clientDAO.getAllClients();  }    public boolean addClaim(Claim claim) {  return claimDAO.addClaim(claim);  }  public boolean updateClaimStatus(int claimId, String status) {  return claimDAO.updateClaimStatus(claimId, status);  }  public Claim getClaimById(int claimId) {  return claimDAO.getClaimById(claimId);  }  public List<Claim> getAllClaims() {  return claimDAO.getAllClaims();  }    public boolean addPayment(Payment payment) {  return paymentDAO.addPayment(payment);  }  public List<Payment> getPaymentsByClientId(int clientId) {  return paymentDAO.getPaymentsByClientId(clientId);  }  } |

**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.**

***DBConnection.java***

|  |
| --- |
| package util;  import java.sql.Connection;  import java.sql.DriverManager;  public class DBConnection {  private static Connection connection;  public static Connection getConnection() {  try {  if (connection == null || connection.isClosed()) {  String url = PropertyUtil.getProperty("db.url");  String username = PropertyUtil.getProperty("db.username");  String password = PropertyUtil.getProperty("db.password");  Class.forName("com.mysql.cj.jdbc.Driver");  connection = DriverManager.getConnection(url, username, password);  }  } catch (Exception e) {  System.out.println("Error connecting to DB: " + e.getMessage());  }  return connection;  }  } |

***PropertyUtil.java***

|  |
| --- |
| package util;  import java.io.FileInputStream;  import java.io.IOException;  import java.util.Properties;  public class PropertyUtil {  public static String getProperty(String key) {  Properties properties = new Properties();  try (FileInputStream fis = new FileInputStream("src/db.properties")) {  properties.load(fis);  return properties.getProperty(key);  } catch (IOException e) {  System.out.println("Failed to load properties file: " + e.getMessage());  }  return null;  }  } |

***db.properties***

|  |
| --- |
| db.url=jdbc:mysql://localhost:3306/insurancemanagement  db.username=root  db.password=cats |

**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**

|  |
| --- |
| **package** exception;  **public** **class** PolicyNotFoundException **extends** Exception {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **public** PolicyNotFoundException(String message) {  **super**(message);  }  } |

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

**Trigger all the methods in service implementation class.**

|  |
| --- |
| **package** main;  **import** entity.\*;  **import** myexceptions.PolicyNotFoundException;  **import** dao.InsuranceServiceImpl;  **import** java.sql.Date;  **import** java.util.List;  **import** java.util.Scanner;  **public** **class** MainModule {  **public** **static** **void** main(String[] args) **throws** PolicyNotFoundException {  Scanner sc = **new** Scanner(System.***in***);  InsuranceServiceImpl service = **new** InsuranceServiceImpl();  **try** {  **if** (service.loginUser("admin", "admin123") == **null**) {  service.registerUser(**new** User(0, "admin", "admin123", "admin"));  service.registerUser(**new** User(0, "sam", "sam123", "client"));  }  **if** (service.getAllPolicies().isEmpty()) {  service.addPolicy(**new** Policy(1, "Health Plus", 500000, 1500));  service.addPolicy(**new** Policy(2, "Life Secure", 1000000, 3000));  service.addPolicy(**new** Policy(3, "Vehicle Shield", 200000, 1000));  }  **if** (service.getAllClients().isEmpty()) {  service.addClient(**new** Client(0, "John Doe", "john@example.com", 1));  service.addClient(**new** Client(0, "Alice Smith", "alice@example.com", 2));  }  **if** (service.getAllClaims().isEmpty()) {  service.addClaim(**new** Claim(0, "CLM001", Date.*valueOf*("2024-03-01"), 40000, "Pending", 1, 1));  service.addClaim(**new** Claim(0, "CLM002", Date.*valueOf*("2024-03-05"), 100000, "Approved", 2, 2));  }  service.addPayment(**new** Payment(0, Date.*valueOf*("2024-03-10"), 1500, 1));  service.addPayment(**new** Payment(0, Date.*valueOf*("2024-03-12"), 3000, 2));  } **catch** (Exception e) {  System.***out***.println("Startup error: " + e.getMessage());  }  **while** (**true**) {  System.***out***.println("\n=== Insurance Management System ===");  System.***out***.println("1. Register User");  System.***out***.println("2. Login User");  System.***out***.println("3. Add Policy");  System.***out***.println("4. View All Policies");  System.***out***.println("5. Add Client");  System.***out***.println("6. File Claim");  System.***out***.println("7. Make Payment");  System.***out***.println("8. Exit");  System.***out***.print("Choose an option: ");  **int** choice = sc.nextInt();  sc.nextLine();  **switch** (choice) {  **case** 1 -> {  System.***out***.print("Username: ");  String uname = sc.nextLine();  System.***out***.print("Password: ");  String pass = sc.nextLine();  System.***out***.print("Role: ");  String role = sc.nextLine();  User u = **new** User(0, uname, pass, role);  **boolean** added = service.registerUser(u);  System.***out***.println(added ? "User registered." : "Registration failed.");  }  **case** 2 -> {  System.***out***.print("Username: ");  String uname = sc.nextLine();  System.***out***.print("Password: ");  String pass = sc.nextLine();  User user = service.loginUser(uname, pass);  System.***out***.println(user != **null** ? "Welcome " + user.getUsername() : "Login failed.");  }  **case** 3 -> {  System.***out***.print("Policy ID: ");  **int** policyId = sc.nextInt();  sc.nextLine();  **if** (policyId != 1 && policyId != 2 && policyId != 3) {  **throw** **new** PolicyNotFoundException("Invalid Policy ID: " + policyId);  }  System.***out***.print("Policy Name: ");  String name = sc.nextLine();  System.***out***.print("Coverage Amount: ");  **double** coverage = sc.nextDouble();  System.***out***.print("Premium: ");  **double** premium = sc.nextDouble();  Policy p = **new** Policy(policyId, name, coverage, premium);  **if** (!service.addPolicy(p)) {  **throw** **new** PolicyNotFoundException("Failed to insert policy: " + name);  }  System.***out***.println("Policy added successfully.");  }  **case** 4 -> {  List<Policy> policies = service.getAllPolicies();  policies.forEach(System.***out***::println);  }  **case** 5 -> {  System.***out***.print("Client Name: ");  String cname = sc.nextLine();  System.***out***.print("Contact Info: ");  String contact = sc.nextLine();  System.***out***.print("Policy ID: ");  **int** pid = sc.nextInt();  Client client = **new** Client(0, cname, contact, pid);  **boolean** added = service.addClient(client);  System.***out***.println(added ? "Client added successfully." : "Failed to add client.");  }  **case** 6 -> {  System.***out***.print("Claim Number: ");  String cno = sc.nextLine();  System.***out***.print("Claim Amount: ");  **double** amount = sc.nextDouble();  sc.nextLine();  System.***out***.print("Status: ");  String status = sc.nextLine();  System.***out***.print("Policy ID: ");  **int** pid = sc.nextInt();  System.***out***.print("Client ID: ");  **int** cid = sc.nextInt();  Claim claim = **new** Claim(0, cno, **new** Date(System.*currentTimeMillis*()), amount, status, pid, cid);  **boolean** added = service.addClaim(claim);  System.***out***.println(added ? "Claim filed." : "Failed to file claim.");  }  **case** 7 -> {  System.***out***.print("Payment Amount: ");  **double** amt = sc.nextDouble();  System.***out***.print("Client ID: ");  **int** cid = sc.nextInt();  Payment payment = **new** Payment(0, **new** Date(System.*currentTimeMillis*()), amt, cid);  **boolean** added = service.addPayment(payment);  System.***out***.println(added ? "Payment successful." : "Failed to add payment.");  }  **case** 8 -> {  System.***out***.println("Exiting system.");  sc.close();  System.*exit*(0);  }  **default** -> System.***out***.println("Invalid option.");  }  }  }  } |

***OUTPUT:***

|  |
| --- |
| === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 1  Username: sangeetha  Password: 123456  Role: client  User registered.  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 2  Username: sangeetha  Password: 123456  Welcome sangeetha  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 3  Policy Name: Health Plus  Coverage Amount: 1200000  Premium: 12300  Policy added.  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 4  Policy [policyId=1, policyName=Health Plus, coverageAmount=500000.0, premium=1500.0]  Policy [policyId=2, policyName=Life Secure, coverageAmount=1000000.0, premium=3000.0]  Policy [policyId=3, policyName=Vehicle Shield, coverageAmount=200000.0, premium=1000.0]  Policy [policyId=4, policyName=Health Plus, coverageAmount=1200000.0, premium=12300.0]  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 5  Client Name: saujanya  Contact Info: saujanya@gmail.com  Policy ID: 3  Client added.  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 6  Claim Number: CLM003  Claim Amount: 20000  Status: Pending  Policy ID: 3  Client ID: 3  Claim filed.  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 7  Payment Amount: 120000  Client ID: 3  Payment done.  === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 8  Exiting system. |

***EXCEPTION OUTPUT:***

|  |
| --- |
| === Insurance Management System ===  1. Register User  2. Login User  3. Add Policy  4. View All Policies  5. Add Client  6. File Claim  7. Make Payment  8. Exit  Choose an option: 3  Policy ID: 5  Exception in thread "main" myexceptions.PolicyNotFoundException: Invalid Policy ID: 5  at main.MainModule.main(MainModule.java:87) |