Hexaware Java Coding Challenge

Order Management System

Submitted by – Mukund Suresh Sutar

Email – mukundmoto89@gmail.com

GitHub Link-
https://github.com/mukkund05/OrderManagementSystem

Create SQL Schema from the product and user class, use the class attributes for table column names.

```sql
CREATE TABLE Products (
    productId INT AUTO_INCREMENT PRIMARY KEY,
    productName VARCHAR(50) NOT NULL,
    description TEXT,
    price DOUBLE,
    quantityInStock INT,
    type ENUM('Electronics', 'Clothing'),
    brand VARCHAR(200),
    warrantyPeriod INT,
    size VARCHAR(10),
    color VARCHAR(50)
);
```

```sql
CREATE TABLE Users (
    userId INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('Admin', 'User') NOT NULL
);
```

```sql
CREATE TABLE Orders (
    orderId INT AUTO_INCREMENT PRIMARY KEY,
    userId INT NOT NULL,
    productId INT NOT NULL,
    orderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (userId) REFERENCES Users(userId) ON DELETE CASCADE,
    FOREIGN KEY (productId) REFERENCES Products(productId) ON DELETE CASCADE
);
```

Create a base class called **Product**

```java
package com.hexaware.entity;

public class Product {

    private int productId;
    private String productName;
    private String description;
    private double price;
    private int quantityInStock;
    private Type type;



    public enum Type {
        ELECTRONICS, CLOTHING;
    }

    // Constructor
    public Product(int productId, String productName, String description, double price, int quantityInStock, Type type) {
        this.productId = productId;
        this.productName = productName;
        this.description = description;
        this.price = price;
        this.quantityInStock = quantityInStock;
        this.type = type;
    }

    // Getters and Setters
    public int getProductId() {
        return productId;
    }

    public void setProductId(int productId) {
        this.productId = productId;
    }

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }
```

```java
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getQuantityInStock() {
        return quantityInStock;
    }

    public void setQuantityInStock(int quantityInStock) {
        this.quantityInStock = quantityInStock;
    }

    public Type getType() {
        return type;
    }

    public void setType(Type type) {
        this.type = type;
    }
```

Create a subclass **Electronics** that inherits from Product.

```java
package com.hexaware.entity;

public class Electronics extends Product{


    private String brand;
    private int warrantyPeriod;


    public Electronics(int productId, String productName, String description, double price, int quantityInStock, Type type, String brand, int warrantyPeriod) {
        super(productId, productName, description, price, quantityInStock, type);
        this.brand = brand;
        this.warrantyPeriod = warrantyPeriod;
    }

    public Electronics(String productName, String description, double price, int quantityInStock, Type type, String brand, int warrantyPeriod) {
        super(0, productName, description, price, quantityInStock, type); // 0 as placeholder
        this.brand = brand;
        this.warrantyPeriod = warrantyPeriod;
    }



    public String getBrand() {
        return brand;
    }


    public void setBrand(String brand) {
        this.brand = brand;
    }


    public int getWarrantyPeriod() {
        return warrantyPeriod;
    }


    public void setWarrantyPeriod(int warrantyPeriod) {
        this.warrantyPeriod = warrantyPeriod;
    }
```

Create a subclass **Clothing** that also inherits from Product

```java
package com.hexaware.entity;

public class Clothing extends Product {

    private String size;
    private String color;

    // Constructor
    public Clothing(int productId, String productName, String description, double price, int quantityInStock, Type type, String size, String color) {
        super(productId, productName, description, price, quantityInStock, type);
        this.size = size;
        this.color = color;
    }

    public Clothing(String productName, String description, double price, int quantityInStock, Type type, String size, String color) {
        super(0, productName, description, price, quantityInStock, type);
        this.size = size;
        this.color = color;
    }


    // Getters and Setters
    public String getSize() {
        return size;
    }

    public void setSize(String size) {
        this.size = size;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

Create a **User** class with attributes:

```java
package com.hexaware.entity;

public class User {

    private int userId;
    private String username;
    private String password;
    private Role role;

    public enum Role{
        Admin,User
    }

    public User(String username, String password, Role role){

        this.username = username;
        this.password = password;
        this.role = role;
    }

    // Getter and Setter

    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 Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    };
```

Define an interface/abstract class named **IOrderManagementRepository**

```java
package com.hexaware.dao;

import com.hexaware.entity.*;
import java.util.List;

public interface IOrderManagementRepository {

    void createOrder(User user, List<Product> products);
    void cancelOrder(int userId, int orderId);
    void createProduct(User user, Product product);
    void createUser(User user);
    List<Product> getAllProducts();
    List<Product> getOrderByUser(User user);
}
```

Implement the IOrderManagementRepository interface/abstractclass in a class called **OrderProcessor**

```java
package com.hexaware.dao;

import com.hexaware.entity.*;
import com.hexaware.util.*;
import com.hexaware.exceptions.*;

import java.sql.*;
import java.util.*;

public class OrderProcessor implements IOrderManagementRepository {
    private Connection connection;

    public OrderProcessor() {
        String propertyFileName = "E:\\JAVA\\Order Management\\OrderManagementSystem\\src\\com\\hexaware\\util\\db.properties";
        this.connection = DBConnUtil.getConnection(propertyFileName);
    }

    @Override
    public void createOrder(User user, List<Product> products) {
        try (Connection conn = DBUtil.getDBConn()) {

            if (!userExists(user.getUsername(), conn)) {
                createUser(user);
            }


            int userId;
            try {
                userId = getUserIdByUsername(user.getUsername(), conn);
            } catch (UserNotFoundException e) {
                System.out.println("Error: Failed to retrieve user ID after creation: " + e.getMessage());
                return;
            }

            // Check if products exist
            for (Product product : products) {
                if (!productExists(product.getProductId(), conn)) {
                    throw new ProductNotFoundException("Product ID " + product.getProductId() + " not found.");
                }
            }

            // Insert order
            for (Product product : products) {
```

```
            // Insert order
            for (Product product : products) {
                String query = "INSERT INTO Orders(userId, productId) VALUES(?, ?)";
                try (PreparedStatement ptst = conn.prepareStatement(query)) {
                    ptst.setInt(1, userId);
                    ptst.setInt(2, product.getProductId());
                    ptst.executeUpdate();
                }
            }

            System.out.println("Order created successfully.");

        } catch (SQLException e) {
            System.out.println("Database error in creating order: " + e.getMessage());
        } catch (ProductNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void cancelOrder(int userId, int orderId) {
        try (Connection conn = DBUtil.getDBConn()) {
            // Check if user exists
            if (!userExists(userId, conn)) {
                throw new UserNotFoundException("User ID " + userId + " not found.");
            }

            // Check if order exists
            String checkQuery = "SELECT * FROM Orders WHERE orderId=? AND userId=?";
            try (PreparedStatement checkStmt = conn.prepareStatement(checkQuery)) {
                checkStmt.setInt(1, orderId);
                checkStmt.setInt(2, userId);
                ResultSet rs = checkStmt.executeQuery();
                if (!rs.next()) {
                    throw new OrderNotFoundException("Order ID " + orderId + " not found for User ID " + userId);
                }
            }

            // Delete order
            String deleteQuery = "DELETE FROM Orders WHERE orderId=?";
            try (PreparedStatement deleteStmt = conn.prepareStatement(deleteQuery)) {
                deleteStmt.setInt(1, orderId);
```

```
                deleteStmt.executeUpdate();
                System.out.println("Order cancelled successfully.");
            }

        } catch (SQLException e) {
            System.out.println("Database error in cancelling order: " + e.getMessage());
        } catch (UserNotFoundException | OrderNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    @Override
    public void createProduct(User user, Product product) {
        try (Connection conn = DBUtil.getDBConn()) {
            // Verify user exists and is Admin
            if (!userExists(user.getUsername(), conn)) {
                throw new UserNotFoundException("User " + user.getUsername() + " not found.");
            }
            String roleQuery = "SELECT role FROM Users WHERE username=?";
            try (PreparedStatement roleStmt = conn.prepareStatement(roleQuery)) {
                roleStmt.setString(1, user.getUsername());
                ResultSet rs = roleStmt.executeQuery();
                if (rs.next() && !rs.getString("role").equals("Admin")) {
                    throw new IllegalAccessException("Only Admin users can add products.");
                }
            }

            // Insert product
            String query = "INSERT INTO Products(productName, description, price, quantityInStock, type, brand, warrantyPeriod, size, color) VALUES(?,?,?,?,?,?,?,?,?)";
            try (PreparedStatement ptst = conn.prepareStatement(query)) {
                ptst.setString(1, product.getProductName());
                ptst.setString(2, product.getDescription());
                ptst.setDouble(3, product.getPrice());
                ptst.setInt(4, product.getQuantityInStock());
                ptst.setString(5, product.getType().toString());

                if (product instanceof Electronics) {
                    ptst.setString(6, ((Electronics) product).getBrand());
                    ptst.setInt(7, ((Electronics) product).getWarrantyPeriod());
                    ptst.setNull(8, Types.VARCHAR);
                    ptst.setNull(9, Types.VARCHAR);
```

```
                ptst.setInt(7, ((Electronics) product).getWarrantyPeriod());
                ptst.setNull(8, Types.VARCHAR);
                ptst.setNull(9, Types.VARCHAR);
            } else if (product instanceof Clothing) {
                ptst.setNull(6, Types.VARCHAR);
                ptst.setNull(7, Types.INTEGER);
                ptst.setString(8, ((Clothing) product).getSize());
                ptst.setString(9, ((Clothing) product).getColor());
            } else {
                ptst.setNull(6, Types.VARCHAR);
                ptst.setNull(7, Types.INTEGER);
                ptst.setNull(8, Types.VARCHAR);
                ptst.setNull(9, Types.VARCHAR);
            }

            ptst.executeUpdate();
            System.out.println("Product created successfully!");
        }

    } catch (SQLException e) {
        System.out.println("Database error in creating product: " + e.getMessage());
    } catch (UserNotFoundException | IllegalAccessException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

@Override
public void createUser(User user) {
    try (Connection conn = DBUtil.getDBConn()) {
        if (userExists(user.getUsername(), conn)) {
            throw new IllegalArgumentException("User " + user.getUsername() + " already exists.");
        }

        String query = "INSERT INTO Users(username, password, role) VALUES(?,?,?)";
        try (PreparedStatement ptst = conn.prepareStatement(query)) {
            ptst.setString(1, user.getUsername());
            ptst.setString(2, user.getPassword());
            ptst.setString(3, user.getRole().toString());
            ptst.executeUpdate();
            System.out.println("User created successfully.");
        }
    } catch (SQLException e) {
```

```
        System.out.println("Database error in creating user: " + e.getMessage());
    } catch (IllegalArgumentException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

@Override
public List<Product> getAllProducts() {
    List<Product> productList = new ArrayList<>();
    String query = "SELECT * FROM Products";

    try (Connection conn = DBUtil.getDBConn(); PreparedStatement ptst = conn.prepareStatement(query)) {
        ResultSet rs = ptst.executeQuery();

        while (rs.next()) {
            int id = rs.getInt("productId");
            String name = rs.getString("productName");
            String desc = rs.getString("description");
            double price = rs.getDouble("price");
            int stock = rs.getInt("quantityInStock");
            Product.Type type = Product.Type.valueOf(rs.getString("type"));

            if (type == Product.Type.ELECTRONICS) {
                String brand = rs.getString("brand");
                int warranty = rs.getInt("warrantyPeriod");
                productList.add(new Electronics(id, name, desc, price, stock, type, brand, warranty));
            } else if (type == Product.Type.CLOTHING) {
                String size = rs.getString("size");
                String color = rs.getString("color");
                productList.add(new Clothing(id, name, desc, price, stock, type, size, color));
            }
        }
    } catch (SQLException e) {
        System.out.println("Database error in fetching products: " + e.getMessage());
    }

    return productList;
}

@Override
public List<Product> getOrderByUser(User user) {
    List<Product> orderedProducts = new ArrayList<>();
```

Exceptions:

OrderNotFoundException

```java
package com.hexaware.exceptions;

public class OrderNotFoundException extends Exception {

    public OrderNotFoundException(String message) {
        super(message);
    }
}
```

ProductNotFoundException

```java
package com.hexaware.exceptions;

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

UserNotFoundException

```java
package com.hexaware.exceptions;

public class UserNotFoundException extends Exception {

    public UserNotFoundException(String message) {
        super(message);
    }
}
```

Create DBUtil class and add the following method. • static getDBConn():Connection Establish a connection to the database and return

DBUtil

```java
package com.hexaware.util;

import java.sql.Connection;
import java.sql.SQLException;

public class DBUtil {
    public static Connection getDBConn() throws SQLException {
        String propertyFileName = "E:\\JAVA\\Order Management\\OrderManagementSystem\\src\\com\\hexaware\\util\\db.properties";
        return DBConnUtil.getConnection(propertyFileName);
    }
}
```

DBConnUtil

```java
package com.hexaware.util;

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

public class DBConnUtil {

    public static Connection getConnection(String propertyFileName) {
        String connectionString = DBPropertyUtil.getConnectionString(propertyFileName);
        if (connectionString == null) {
            System.err.println("Unable to fetch connection string.");
            return null;
        }

        try {
            Connection connection = DriverManager.getConnection(connectionString);
            return connection;
        } catch (SQLException e) {
            System.err.println("Error establishing connection: " + e.getMessage());
            return null;
        }
    }
}
```

DBPropertyUtil

```java
package com.hexaware.util;

import java.io.FileInputStream;

public class DBPropertyUtil {

    public static String getConnectionString(String propertyFileName) {

        Properties properties = new Properties();

        try (FileInputStream fis = new FileInputStream(propertyFileName)) {

            properties.load(fis);

            String url = properties.getProperty("db.url");
            String user = properties.getProperty("db.user");
            String password = properties.getProperty("db.password");

            if (url != null && user != null && password != null) {
                return url + "?user=" + user + "&password=" + password;
            } else {
                throw new IllegalArgumentException("Missing required database properties!");
            }
        } catch (IOException | IllegalArgumentException e) {
            System.err.println("Error reading or constructing connection string: " + e.getMessage());
            return null;
        }
    }
}
```

Create **OrderManagement** main class and perform following operation:

```java
package com.hexaware.main;

import com.hexaware.dao.OrderProcessor;

public class OrderManagement  {
    static Scanner scanner = new Scanner(System.in);
    static OrderProcessor orderProcessor = new OrderProcessor();

    public static void main(String[] args) {
        while (true) {
            System.out.println("\n--- Order Management System ---");
            System.out.println("1. Create User");
            System.out.println("2. Create Product");
            System.out.println("3. Create Order");
            System.out.println("4. Cancel Order");
            System.out.println("5. Get All Products");
            System.out.println("6. Get Orders by User");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");
            try {
                int choice = scanner.nextInt();
                scanner.nextLine();

                switch (choice) {
                    case 1 -> createUser();
                    case 2 -> createProduct();
                    case 3 -> createOrder();
                    case 4 -> cancelOrder();
                    case 5 -> getAllProducts();
                    case 6 -> getOrderByUser();
                    case 7 -> {
                        System.out.println("Exiting...");
                        System.exit(0);
                    }
                    default -> System.out.println("Invalid choice. Please enter a number between 1 and 7.");
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input. Please enter a number.");
                scanner.nextLine();
            }
        }
    }
```

```java
private static void createUser() {
    System.out.print("Enter username: ");
    String username = scanner.nextLine();
    System.out.print("Enter password: ");
    String password = scanner.nextLine();
    System.out.print("Role (Admin/User): ");
    String role = scanner.nextLine().trim();
    try {
        User.Role userRole = User.Role.valueOf(role);
        User user = new User(username, password, userRole);
        orderProcessor.createUser(user);
    } catch (IllegalArgumentException e) {
        System.out.println("Invalid role. Please enter 'Admin' or 'User'.");
    }
}

private static void createProduct() {
    System.out.print("Admin Username: ");
    String username = scanner.nextLine();
    System.out.print("Admin Password: ");
    String password = scanner.nextLine();
    User user = new User(username, password, User.Role.Admin);

    System.out.print("Product Name: ");
    String name = scanner.nextLine();
    System.out.print("Description: ");
    String desc = scanner.nextLine();
    System.out.print("Price: ");
    double price;
    try {
        price = scanner.nextDouble();
    } catch (InputMismatchException e) {
        System.out.println("Invalid price. Please enter a number.");
        scanner.nextLine();
        return;
    }
    System.out.print("Stock: ");
    int stock;
    try {
        stock = scanner.nextInt();
    } catch (InputMismatchException e) {
        System.out.println("Invalid stock. Please enter an integer.");
```

```java
            scanner.nextLine();
            return;
        }
        scanner.nextLine();
        System.out.print("Type (ELECTRONICS/CLOTHING): ");
        String type = scanner.nextLine().trim().toUpperCase();

        Product product = null;
        try {
            Product.Type productType = Product.Type.valueOf(type);
            if (productType == Product.Type.ELECTRONICS) {
                System.out.print("Brand: ");
                String brand = scanner.nextLine();
                System.out.print("Warranty (months): ");
                int warranty;
                try {
                    warranty = scanner.nextInt();
                } catch (InputMismatchException e) {
                    System.out.println("Invalid warranty. Please enter an integer.");
                    scanner.nextLine();
                    return;
                }
                scanner.nextLine();
                product = new Electronics(name, desc, price, stock, productType, brand, warranty);
            } else if (productType == Product.Type.CLOTHING) {
                System.out.print("Size: ");
                String size = scanner.nextLine();
                System.out.print("Color: ");
                String color = scanner.nextLine();
                product = new Clothing(name, desc, price, stock, productType, size, color);
            }
            orderProcessor.createProduct(user, product);
        } catch (IllegalArgumentException e) {
            System.out.println("Invalid product type. Please enter 'ELECTRONICS' or 'CLOTHING'.");
        }
    }
```

```java
private static void createOrder() {
    System.out.print("Username: ");
    String username = scanner.nextLine();
    System.out.print("Password: ");
    String password = scanner.nextLine();
    User user = new User(username, password, User.Role.User);

    List<Product> allProducts = orderProcessor.getAllProducts();
    if (allProducts.isEmpty()) {
        System.out.println("No products available.");
        return;
    }

    System.out.println("Available Products:");
    for (Product p : allProducts) {
        System.out.println(p.getProductId() + ": " + p.getProductName());
    }

    System.out.print("Enter Product IDs (comma separated): ");
    String[] ids = scanner.nextLine().split(",");
    List<Product> orderList = new ArrayList<>();
    try {
        for (String id : ids) {
            int pid = Integer.parseInt(id.trim());
            Optional<Product> product = allProducts.stream().filter(p -> p.getProductId() == pid).findFirst();
            if (product.isPresent()) {
                orderList.add(product.get());
            } else {
                System.out.println("Product ID " + pid + " not found.");
                return;
            }
        }
        orderProcessor.createOrder(user, orderList);
    } catch (NumberFormatException e) {
        System.out.println("Invalid product ID format. Please enter valid numbers.");
    }
}
```

```java
private static void cancelOrder() {
    System.out.print("Enter your User ID: ");
    int userId;
    try {
        userId = scanner.nextInt();
    } catch (InputMismatchException e) {
        System.out.println("Invalid User ID. Please enter an integer.");
        scanner.nextLine();
        return;
    }
    System.out.print("Enter Order ID to cancel: ");
    int orderId;
    try {
        orderId = scanner.nextInt();
    } catch (InputMismatchException e) {
        System.out.println("Invalid Order ID. Please enter an integer.");
        scanner.nextLine();
        return;
    }
    scanner.nextLine();
    orderProcessor.cancelOrder(userId, orderId);
}

private static void getAllProducts() {
    List<Product> products = orderProcessor.getAllProducts();
    if (products.isEmpty()) {
        System.out.println("No products found.");
    } else {
        for (Product p : products) {
            System.out.println(p.getProductId() + ": " + p.getProductName() + " - " + p.getType());
        }
    }
}
```

```
    private static void getOrderByUser() {
        System.out.print("Enter Username: ");
        String username = scanner.nextLine();
        System.out.print("Password: ");
        String password = scanner.nextLine();
        User user = new User(username, password, User.Role.User);
        List<Product> orders = orderProcessor.getOrderByUser(user);
        if (orders.isEmpty()) {
            System.out.println("No orders found for user " + username);
        } else {
            for (Product p : orders) {
                System.out.println(p.getProductId() + ": " + p.getProductName());
            }
        }
    }
}
```

Output:

```
--- Order Management System ---
1. Create User
2. Create Product
3. Create Order
4. Cancel Order
5. Get All Products
6. Get Orders by User
7. Exit
Enter your choice:
```

1. Create User

```
Enter your choice: 1
Enter username: John
Enter password: 123
Role (Admin/User): Admin
User created successfully.
```

2. Create Product

```
Enter your choice: 2
Admin Username: John
Admin Password: 123
Product Name: OnePlus 13R
Description: Mobile Phone
Price: 50000
Stock: 10
Type (ELECTRONICS/CLOTHING): ELECTRONICS
Brand: OnePlus
Warranty (months): 12
Product created successfully!
```

3. Create Order

```
Enter your choice: 3
Username: John
Password: 123
Available Products:
1: OnePlus 12R
2: OnePlus 13R
Enter Product IDs (comma separated): 2
Order created successfully.
```

4. Cancel Order

```
Enter your choice: 4
Enter your User ID: 2
Enter Order ID to cancel: 2
Order cancelled successfully.
```

5. Get All Products

```
Enter your choice: 5
1: OnePlus 12R - ELECTRONICS
2: OnePlus 13R - ELECTRONICS
```

6. Get Orders by User

```
Enter your choice: 6
Enter Username: John
Password: 123
No orders found for user John
```

7. Exit

```
--- Order Management System ---
1. Create User
2. Create Product
3. Create Order
4. Cancel Order
5. Get All Products
6. Get Orders by User
7. Exit
Enter your choice: 7
Exiting...
```