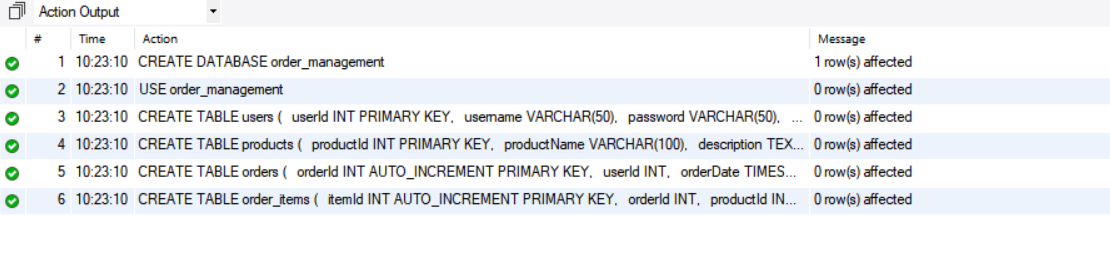
Java Coding Challenge - Order Management System

(Sivaganesh Natarajavel)

SQL Tables:

|  |
| --- |
| CREATE DATABASE order\_management;  USE order\_management;  CREATE TABLE users (  userId INT PRIMARY KEY,  username VARCHAR(50),  password VARCHAR(50),  role VARCHAR(10)  );  CREATE TABLE products (  productId INT PRIMARY KEY,  productName VARCHAR(100),  description TEXT,  price DOUBLE,  quantityInStock INT,  type VARCHAR(20),  brand VARCHAR(50),  warrantyPeriod INT,  size VARCHAR(10),  color VARCHAR(20)  );  CREATE TABLE orders (  orderId INT AUTO\_INCREMENT PRIMARY KEY,  userId INT,  orderDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (userId) REFERENCES users(userId)  );  CREATE TABLE order\_items (  itemId INT AUTO\_INCREMENT PRIMARY KEY,  orderId INT,  productId INT,  quantity INT,  FOREIGN KEY (orderId) REFERENCES orders(orderId),  FOREIGN KEY (productId) REFERENCES products(productId)  ); |

Output:



Java Code:

Entity/Product.java:

|  |
| --- |
| package org.example.entity;  public class Product {  private int productId;  private String productName;  private String description;  private double price;  private int quantityInStock;  private String type;   public Product() {}   public Product(int productId, String productName, String description, double price,  int quantityInStock, String type) {  this.productId = productId;  this.productName = productName;  this.description = description;  this.price = price;  this.quantityInStock = quantityInStock;  this.type = type;  }   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;  }   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 String getType() {  return type;  }   public void setType(String type) {  this.type = type;  } } |

Entity/Electroics.java:

|  |
| --- |
| package org.example.entity;  public class Electronics extends Product {  private String brand;  private int warrantyPeriod;   public Electronics(int productId, String productName, String description, double price,  int quantityInStock, String type, String brand, int warrantyPeriod) {  super(productId, productName, description, price, quantityInStock, type);  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;  } } |

Entity/Clothing.java:

|  |
| --- |
| package org.example.entity;  public class Clothing extends Product {  private String size;  private String color;   public Clothing(int productId, String productName, String description, double price,  int quantityInStock, String type, String size, String color) {  super(productId, productName, description, price, quantityInStock, type);  this.size = size;  this.color = color;  }   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;  } } |

Entity/User.java:

|  |
| --- |
| package org.example.entity;  public class User {  private int userId;  private String username;  private String password;  private String role;   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;  } } |

Dao/IOrderManagementRepository.java:

|  |
| --- |
| package org.example.dao;  import org.example.entity.Product; import org.example.entity.User;  import java.util.List;  public interface IOrderManagementRepository {  void createUser(User user);  void createProduct(User user, Product product);  void createOrder(User user, List<Product> products);  void cancelOrder(int userId, int orderId);  List<Product> getAllProducts();  List<Product> getOrderByUser(User user); } |

Dao/OrderProcessor.java:

|  |
| --- |
| package org.example.dao;  import org.example.entity.Clothing; import org.example.entity.Electronics; import org.example.entity.Product; import org.example.entity.User; import org.example.exception.OrderNotFoundException; import org.example.util.DBConnUtil;  import java.sql.\*; import java.util.ArrayList; import java.util.List;  public class OrderProcessor implements IOrderManagementRepository {   private static final String *DB\_PROP\_FILE* = "src/main/resources/db.properties";   @Override  public void createUser(User user) {  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);  String query = "INSERT INTO users VALUES (?, ?, ?, ?)";  PreparedStatement stmt = conn.prepareStatement(query);  stmt.setInt(1, user.getUserId());  stmt.setString(2, user.getUsername());  stmt.setString(3, user.getPassword());  stmt.setString(4, user.getRole());  stmt.executeUpdate();  conn.close();  } catch (Exception e) {  System.*out*.println("Error in createUser: " + e.getMessage());  }  }   @Override  public void createProduct(User user, Product product) {  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);   // Check if user is admin  String checkQuery = "SELECT role FROM users WHERE userId = ?";  PreparedStatement checkStmt = conn.prepareStatement(checkQuery);  checkStmt.setInt(1, user.getUserId());  ResultSet rs = checkStmt.executeQuery();   if (!rs.next() || !rs.getString("role").equalsIgnoreCase("Admin")) {  System.*out*.println("Only Admin can add products.");  conn.close();  return;  }   String query = "INSERT INTO products VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";  PreparedStatement stmt = conn.prepareStatement(query);  stmt.setInt(1, product.getProductId());  stmt.setString(2, product.getProductName());  stmt.setString(3, product.getDescription());  stmt.setDouble(4, product.getPrice());  stmt.setInt(5, product.getQuantityInStock());  stmt.setString(6, product.getType());   if (product.getType().equalsIgnoreCase("Electronics")) {  Electronics e = (Electronics) product;  stmt.setString(7, e.getBrand());  stmt.setInt(8, e.getWarrantyPeriod());  stmt.setString(9, null);  stmt.setString(10, null);  } else if (product.getType().equalsIgnoreCase("Clothing")) {  Clothing c = (Clothing) product;  stmt.setString(7, null);  stmt.setInt(8, 0);  stmt.setString(9, c.getSize());  stmt.setString(10, c.getColor());  } else {  stmt.setString(7, null);  stmt.setInt(8, 0);  stmt.setString(9, null);  stmt.setString(10, null);  }   stmt.executeUpdate();  conn.close();  } catch (Exception e) {  System.*out*.println("Error in createProduct: " + e.getMessage());  }  }   @Override  public void createOrder(User user, List<Product> products) {  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);   String checkUser = "SELECT \* FROM users WHERE userId = ?";  PreparedStatement userStmt = conn.prepareStatement(checkUser);  userStmt.setInt(1, user.getUserId());  ResultSet rs = userStmt.executeQuery();   if (!rs.next()) {  createUser(user);  }   String orderQuery = "INSERT INTO orders(userId) VALUES (?)";  PreparedStatement orderStmt = conn.prepareStatement(orderQuery, Statement.*RETURN\_GENERATED\_KEYS*);  orderStmt.setInt(1, user.getUserId());  orderStmt.executeUpdate();   ResultSet generated = orderStmt.getGeneratedKeys();  generated.next();  int orderId = generated.getInt(1);   for (Product p : products) {  String itemQuery = "INSERT INTO order\_items(orderId, productId, quantity) VALUES (?, ?, ?)";  PreparedStatement itemStmt = conn.prepareStatement(itemQuery);  itemStmt.setInt(1, orderId);  itemStmt.setInt(2, p.getProductId());  itemStmt.setInt(3, 1);  itemStmt.executeUpdate();  }   conn.close();  } catch (Exception e) {  System.*out*.println("Error in createOrder: " + e.getMessage());  }  }   @Override  public void cancelOrder(int userId, int orderId) {  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);   String checkQuery = "SELECT \* FROM orders WHERE orderId = ? AND userId = ?";  PreparedStatement checkStmt = conn.prepareStatement(checkQuery);  checkStmt.setInt(1, orderId);  checkStmt.setInt(2, userId);  ResultSet rs = checkStmt.executeQuery();   if (!rs.next()) {  throw new OrderNotFoundException("Order not found.");  }   String deleteItems = "DELETE FROM order\_items WHERE orderId = ?";  PreparedStatement delItemStmt = conn.prepareStatement(deleteItems);  delItemStmt.setInt(1, orderId);  delItemStmt.executeUpdate();   String deleteOrder = "DELETE FROM orders WHERE orderId = ?";  PreparedStatement delOrderStmt = conn.prepareStatement(deleteOrder);  delOrderStmt.setInt(1, orderId);  delOrderStmt.executeUpdate();   conn.close();  } catch (OrderNotFoundException e) {  System.*out*.println(e.getMessage());  } catch (Exception e) {  System.*out*.println("Error in cancelOrder: " + e.getMessage());  }  }   @Override  public List<Product> getAllProducts() {  List<Product> list = new ArrayList<>();  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);  String query = "SELECT \* FROM products";  Statement stmt = conn.createStatement();  ResultSet rs = stmt.executeQuery(query);   while (rs.next()) {  Product p = new Product(  rs.getInt("productId"),  rs.getString("productName"),  rs.getString("description"),  rs.getDouble("price"),  rs.getInt("quantityInStock"),  rs.getString("type")  );  list.add(p);  }   conn.close();  } catch (Exception e) {  System.*out*.println("Error in getAllProducts: " + e.getMessage());  }   return list;  }   @Override  public List<Product> getOrderByUser(User user) {  List<Product> list = new ArrayList<>();  try {  Connection conn = DBConnUtil.*getConnection*(*DB\_PROP\_FILE*);  String query = "SELECT p.\* FROM products p " +  "JOIN order\_items oi ON p.productId = oi.productId " +  "JOIN orders o ON oi.orderId = o.orderId " +  "WHERE o.userId = ?";  PreparedStatement stmt = conn.prepareStatement(query);  stmt.setInt(1, user.getUserId());  ResultSet rs = stmt.executeQuery();   while (rs.next()) {  Product p = new Product(  rs.getInt("productId"),  rs.getString("productName"),  rs.getString("description"),  rs.getDouble("price"),  rs.getInt("quantityInStock"),  rs.getString("type")  );  list.add(p);  }   conn.close();  } catch (Exception e) {  System.*out*.println("Error in getOrderByUser: " + e.getMessage());  }   return list;  } } |

Exception/OrderNotFoundException.java:

|  |
| --- |
| package org.example.exception;  public class OrderNotFoundException extends Exception {  public OrderNotFoundException(String msg) {  super(msg);  } } |

Exception/UserNotFoundException.java:

|  |
| --- |
| package org.example.exception;  public class UserNotFoundException extends Exception {  public UserNotFoundException(String msg) {  super(msg);  } } |

Main/MainModule.java:

|  |
| --- |
| package org.example.main;  import org.example.dao.OrderProcessor; import org.example.entity.\*; import java.util.\*;  public class MainModule {  public static void main(String[] args) {  Scanner sc = new Scanner(System.*in*);  OrderProcessor op = new OrderProcessor(); // Uses DBConnUtil internally   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: ");  int ch = sc.nextInt();  sc.nextLine(); // clear buffer   switch (ch) {  case 1 -> {  System.*out*.print("Enter User ID: ");  int id = sc.nextInt();  sc.nextLine();  System.*out*.print("Enter Username: ");  String name = sc.nextLine();  System.*out*.print("Enter Password: ");  String pass = sc.nextLine();  System.*out*.print("Enter Role (Admin/User): ");  String role = sc.nextLine();   User user = new User(id, name, pass, role);  op.createUser(user);  System.*out*.println("User created successfully.");  }   case 2 -> {  System.*out*.print("Enter Admin User ID: ");  int aid = sc.nextInt();  sc.nextLine();   User admin = new User(aid, "admin", "admin", "Admin");   System.*out*.print("Enter Product ID: ");  int pid = sc.nextInt(); sc.nextLine();  System.*out*.print("Enter Product Name: ");  String pname = sc.nextLine();  System.*out*.print("Enter Description: ");  String desc = sc.nextLine();  System.*out*.print("Enter Price: ");  double price = sc.nextDouble();  System.*out*.print("Enter Quantity in Stock: ");  int qty = sc.nextInt(); sc.nextLine();  System.*out*.print("Enter Type (Electronics/Clothing): ");  String type = sc.nextLine();   Product product;   if (type.equalsIgnoreCase("Electronics")) {  System.*out*.print("Enter Brand: ");  String brand = sc.nextLine();  System.*out*.print("Enter Warranty Period (months): ");  int warranty = sc.nextInt(); sc.nextLine();  product = new Electronics(pid, pname, desc, price, qty, type, brand, warranty);  } else if (type.equalsIgnoreCase("Clothing")) {  System.*out*.print("Enter Size: ");  String size = sc.nextLine();  System.*out*.print("Enter Color: ");  String color = sc.nextLine();  product = new Clothing(pid, pname, desc, price, qty, type, size, color);  } else {  System.*out*.println("Invalid type. Must be 'Electronics' or 'Clothing'.");  break;  }   op.createProduct(admin, product);  System.*out*.println("Product created successfully.");  }   case 3 -> {  System.*out*.print("Enter User ID: ");  int uid = sc.nextInt(); sc.nextLine();  User user = new User(uid, "user", "pass", "User");   List<Product> cart = new ArrayList<>();  System.*out*.print("Enter number of products to order: ");  int count = sc.nextInt();   for (int i = 0; i < count; i++) {  System.*out*.print("Enter Product ID: ");  int pid = sc.nextInt();  cart.add(new Product(pid, "", "", 0.0, 0, ""));  }   op.createOrder(user, cart);  System.*out*.println("Order placed successfully.");  }   case 4 -> {  System.*out*.print("Enter User ID: ");  int uid = sc.nextInt();  System.*out*.print("Enter Order ID to cancel: ");  int oid = sc.nextInt();   op.cancelOrder(uid, oid);  System.*out*.println("Order cancelled successfully.");  }   case 5 -> {  List<Product> products = op.getAllProducts();  System.*out*.println("\nAvailable Products:");  if (products.isEmpty()) {  System.*out*.println("No products found.");  } else {  for (Product p : products) {  System.*out*.printf("ID: %d | Name: %s | ₹%.2f | Type: %s | Stock: %d\n",  p.getProductId(), p.getProductName(), p.getPrice(), p.getType(), p.getQuantityInStock());  }  }  }   case 6 -> {  System.*out*.print("Enter User ID: ");  int uid = sc.nextInt();  User user = new User(uid, "", "", "User");   List<Product> orders = op.getOrderByUser(user);  System.*out*.println("\n🛒 Orders by User:");  if (orders.isEmpty()) {  System.*out*.println("No orders found.");  } else {  for (Product p : orders) {  System.*out*.printf("Product ID: %d | Name: %s | Type: %s\n",  p.getProductId(), p.getProductName(), p.getType());  }  }  }   case 7 -> {  System.*out*.println("Exiting... Bye!");  System.*exit*(0);  }   default -> System.*out*.println("Invalid choice. Please try again.");  }  }  } } |

Util/DBConnUtil.java:

|  |
| --- |
| package org.example.util;  import java.sql.Connection; import java.sql.DriverManager;  public class DBConnUtil {   public static Connection getConnection(String propertyFilePath) {  Connection conn = null;  try {  String connectionUrl = DBPropertyUtil.*getConnectionString*(propertyFilePath);  conn = DriverManager.*getConnection*(connectionUrl);  } catch (Exception e) {  System.*out*.println("Error connecting to DB: " + e.getMessage());  }  return conn;  } } |

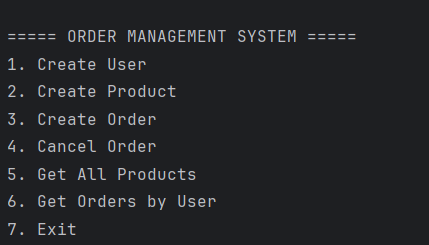
Util/DBPropertyUtil:

|  |
| --- |
| package org.example.util;  import java.io.FileInputStream; import java.io.InputStream; import java.util.Properties;  public class DBPropertyUtil {   public static String getConnectionString(String fileName) {  StringBuilder connStr = new StringBuilder();   try (InputStream input = new FileInputStream(fileName)) {  Properties prop = new Properties();  prop.load(input);   String url = prop.getProperty("db.url");  String user = prop.getProperty("db.user");  String password = prop.getProperty("db.password");   connStr.append(url)  .append("?user=")  .append(user)  .append("&password=")  .append(password);   } catch (Exception e) {  System.*out*.println("Error reading properties file: " + e.getMessage());  }   return connStr.toString();  } } |

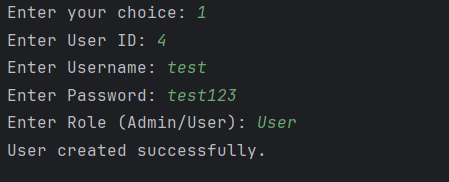
Resources/Db.properties:

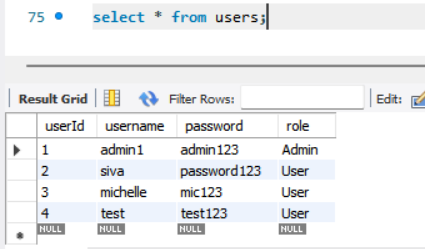
|  |
| --- |
| db.url=jdbc:mysql://localhost:3306/order\_management db.user=root db.password=SivaG@1203 |

Output:

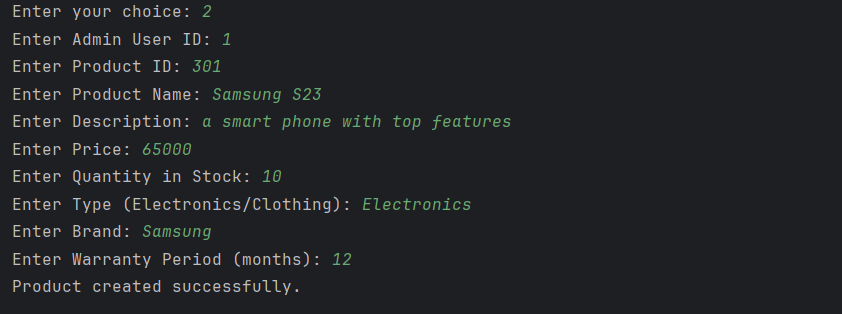


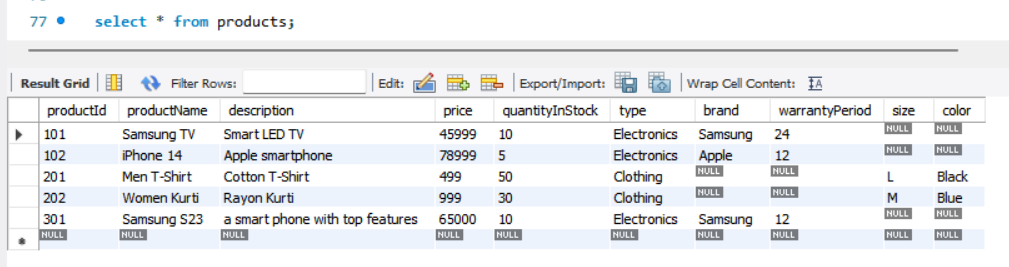
Choice 1:



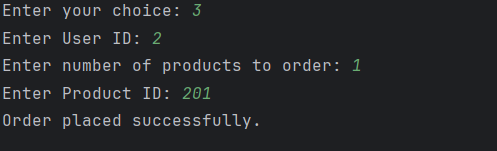


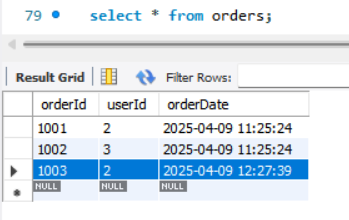
Choice 2:

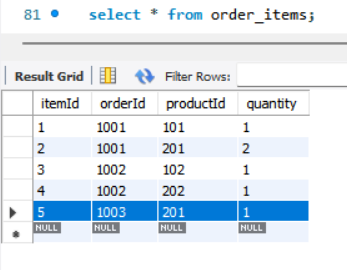




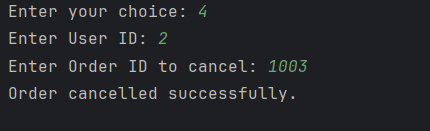
Choice 3:

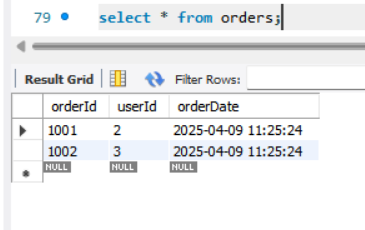


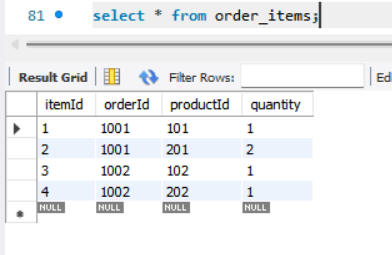




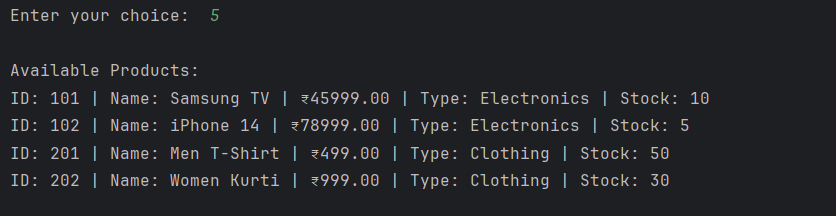
Choice 4:



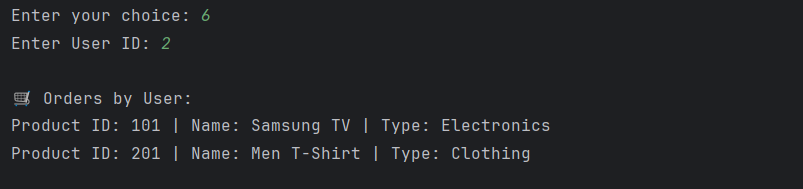




Choice 5:



Choice 6:



Choice 7:

