**CASE STUDY-ECOMMERCE APPLICATION**

DATABASE SCHEMA:

|  |
| --- |
| CREATE DATABASE ECOM;  USE ECOM;  CREATE TABLE customers (  customer\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(100) NOT NULL,  email VARCHAR(255) UNIQUE NOT NULL,  password VARCHAR(255) NOT NULL  );  CREATE TABLE products (  product\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(255) NOT NULL,  price DECIMAL(10,2) NOT NULL,  description TEXT,  quantity INT NOT NULL  );  CREATE TABLE cart (  cart\_id INT PRIMARY KEY AUTO\_INCREMENT,  customer\_id INT NOT NULL,  product\_id INT NOT NULL,  quantity INT NOT NULL CHECK (quantity > 0),  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE,  FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE CASCADE  );  CREATE TABLE orders (  order\_id INT PRIMARY KEY AUTO\_INCREMENT,  customer\_id INT NOT NULL,  order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  total\_price DECIMAL(10,2) NOT NULL CHECK (total\_price >= 0),  shipping\_address TEXT NOT NULL,  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE  );  CREATE TABLE order\_items (  order\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,  order\_id INT NOT NULL,  product\_id INT NOT NULL,  quantity INT NOT NULL CHECK (quantity > 0),  FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE CASCADE,  FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE CASCADE  ); |

|  |
| --- |
| INSERT INTO customers (name, email, password) VALUES  ('Alice', 'alice@example.com', 'pass123'),  ('Bob', 'bob@example.com', 'bobpass'),  ('Charlie', 'charlie@example.com', 'charliepass'),  ('Diana', 'diana@example.com', 'dianapass'),  ('Ethan', 'ethan@example.com', 'ethan123'),  ('Fiona', 'fiona@example.com', 'fiona123'),  ('George', 'george@example.com', 'george123'),  ('Hannah', 'hannah@example.com', 'hannah123'),  ('Irene', 'irene@example.com', 'irene123'),  ('Jack', 'jack@example.com', 'jack123');  INSERT INTO products (name, price, description, quantity) VALUES  ('Wireless Mouse', 499.99, 'Ergonomic wireless mouse', 100),  ('Mechanical Keyboard', 1499.50, 'RGB backlit mechanical keyboard', 80),  ('USB-C Cable', 199.00, 'Fast charging 1m cable', 200),  ('Laptop Stand', 899.00, 'Aluminum adjustable stand', 50),  ('Bluetooth Speaker', 2499.99, 'Portable outdoor speaker', 60),  ('Webcam', 1299.00, '1080p HD webcam', 40),  ('Monitor 24"', 8499.00, '24-inch IPS display', 25),  ('External HDD 1TB', 3499.99, 'USB 3.0 compatible', 35),  ('Wireless Charger', 1099.00, '10W fast charger', 90),  ('Gaming Headset', 2999.00, 'Noise-canceling mic and surround sound', 45);  INSERT INTO cart (customer\_id, product\_id, quantity) VALUES  (1, 1, 2),  (2, 3, 1),  (3, 2, 1),  (4, 4, 1),  (5, 5, 2),  (6, 6, 1),  (7, 7, 1),  (8, 8, 1),  (9, 9, 1),  (10, 10, 2);  INSERT INTO orders (customer\_id, total\_price, shipping\_address) VALUES  (1, 999.98, '101 Main Street'),  (2, 199.00, '202 Oak Avenue'),  (3, 1499.50, '303 Pine Road'),  (4, 899.00, '404 Maple Lane'),  (5, 4999.98, '505 Birch Street'),  (6, 1299.00, '606 Elm Blvd'),  (7, 8499.00, '707 Cedar Drive'),  (8, 3499.99, '808 Walnut Way'),  (9, 1099.00, '909 Poplar Place'),  (10, 5998.00, '1001 Chestnut Court');  INSERT INTO order\_items (order\_id, product\_id, quantity) VALUES  (1, 1, 2),  (2, 3, 1),  (3, 2, 1),  (4, 4, 1),  (5, 5, 2),  (6, 6, 1),  (7, 7, 1),  (8, 8, 1),  (9, 9, 1),  (10, 10, 2); |

OOPS IMPLEMENTATION:

**package entity**

Customer.java

|  |
| --- |
| **package** entity;  **public** **class** Customer {  **private** **int** customerId;  **private** String customername;  **private** String email;  **private** String address;  **private** String password;  **public** Customer(**int** customerId, String customername, String email, String address,String password) {  **this**.customerId = customerId;  **this**.customername = customername;  **this**.email = email;  **this**.address = address;  **this**.setPassword(password);  }    **public** Customer() {}  **public** **int** getCustomerId() {  **return** customerId;  }  **public** **void** setCustomerId(**int** customerId) {  **this**.customerId = customerId;  }  **public** String getCustomerName() {  **return** customername;  }  **public** **void** setName(String name) {  **this**.customername = name;  }  **public** String getEmail() {  **return** email;  }  **public** **void** setEmail(String email) {  **this**.email = email;  }  **public** String getAddress() {  **return** address;  }  **public** **void** setAddress(String address) {  **this**.address = address;  }    @Override  **public** String toString() {  **return** "Customer [ID=" + customerId + ", Name=" + customername + ", Email=" + email + ", Address=" + address + "]";  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  } |

Product.java

|  |
| --- |
| **package** entity;  **public** **class** Product {  **private** **int** productId;  **private** String productname;  **private** **double** price;  **private** **int** quantity;  **private** String description;  **public** Product() {}  **public** Product(**int** productId, String productname, **double** price, String description, **int** quantity) {  **this**.productId = productId;  **this**.productname = productname;  **this**.price = price;  **this**.description = description;  **this**.quantity = quantity;  }  **public** **int** getProductId() { **return** productId; }  **public** **void** setProductId(**int** productId) { **this**.productId = productId; }  **public** String getProductName() { **return** productname; }  **public** **void** setProductName(String name) { **this**.productname = name; }  **public** **double** getPrice() { **return** price; }  **public** **void** setPrice(**double** price) { **this**.price = price; }  **public** **int** getQuantity() { **return** quantity; }  **public** **void** setQuantity(**int** quantity) { **this**.quantity = quantity; }  @Override  **public** String toString() {  **return** "Product [ID=" + productId + ", Name=" + productname + ", Price=" + price + ", Quantity=" + quantity + "]";  }  **public** String getDescription() {  // **TODO** Auto-generated method stub  **return** description;  }  **public** **void** setDescription(String description) {  // **TODO** Auto-generated method stub  **this**.description=description;  }    } |

Order.java

|  |
| --- |
| **package** entity;  **import** java.util.Date;  **public** **class** Order {  **private** **int** orderId;  **private** **int** customerId;  **private** Date orderDate;  **private** **double** totalAmount;  **public** Order() {}  **public** Order(**int** orderId, **int** customerId, Date orderDate, **double** totalAmount) {  **this**.orderId = orderId;  **this**.customerId = customerId;  **this**.orderDate = orderDate;  **this**.totalAmount = totalAmount;  }  **public** **int** getOrderId() { **return** orderId; }  **public** **void** setOrderId(**int** orderId) { **this**.orderId = orderId; }  **public** **int** getCustomerId() { **return** customerId; }  **public** **void** setCustomerId(**int** customerId) { **this**.customerId = customerId; }  **public** Date getOrderDate() { **return** orderDate; }  **public** **void** setOrderDate(Date orderDate) { **this**.orderDate = orderDate; }  **public** **double** getTotalAmount() { **return** totalAmount; }  **public** **void** setTotalAmount(**double** totalAmount) { **this**.totalAmount = totalAmount; }  @Override  **public** String toString() {  **return** "Order [ID=" + orderId + ", CustomerID=" + customerId + ", Date=" + orderDate + ", Total=" + totalAmount + "]";  }  } |

OrderDetails.java

|  |
| --- |
| **package** entity;  **public** **class** OrderDetails {  **private** **int** orderId;  **private** **int** productId;  **private** **int** quantity;  **private** **double** price;  **public** OrderDetails() {}  **public** OrderDetails(**int** orderId, **int** productId, **int** quantity, **double** price) {  **this**.orderId = orderId;  **this**.productId = productId;  **this**.quantity = quantity;  **this**.price = price;  }  **public** **int** getOrderId() { **return** orderId; }  **public** **void** setOrderId(**int** orderId) { **this**.orderId = orderId; }  **public** **int** getProductId() { **return** productId; }  **public** **void** setProductId(**int** productId) { **this**.productId = productId; }  **public** **int** getQuantity() { **return** quantity; }  **public** **void** setQuantity(**int** quantity) { **this**.quantity = quantity; }  **public** **double** getPrice() { **return** price; }  **public** **void** setPrice(**double** price) { **this**.price = price; }  @Override  **public** String toString() {  **return** "OrderDetails [OrderID=" + orderId + ", ProductID=" + productId + ", Quantity=" + quantity + ", Price=" + price + "]";  }  } |

Cart.java

|  |
| --- |
| **package** entity;  **import** java.util.HashMap;  **import** java.util.Map;  **public** **class** Cart {  **private** **int** customerId;  **private** Map<Product, Integer> items = **new** HashMap<>();  **public** Cart(**int** customerId) {  **this**.customerId = customerId;  }  **public** **int** getCustomerId() { **return** customerId; }  **public** Map<Product, Integer> getItems() { **return** items; }  **public** **void** setCustomerId(**int** customerId) {  **this**.customerId = customerId;  }  **public** **void** setItems(Map<Product, Integer> items) {  **this**.items = items;  }  **public** **void** addProduct(Product product, **int** quantity) {  items.put(product, items.getOrDefault(product, 0) + quantity);  }  **public** **void** removeProduct(Product product) {  items.remove(product);  }  **public** **double** calculateTotal() {  **return** items.entrySet().stream()  .mapToDouble(entry -> entry.getKey().getPrice() \* entry.getValue())  .sum();  }  @Override  **public** String toString() {  **return** "Cart [Customer ID=" + customerId + ", Items=" + items + "]";  }  } |

package dao

OrderProcessorRepository.java

|  |
| --- |
| **package** dao;  **import** entity.\*;  **import** exception.\*;  **import** java.util.\*;  **public** **interface** OrderProcessorRepository {  **boolean** createProduct(Product product);  **boolean** createCustomer(Customer customer);  **boolean** deleteProduct(**int** productId);  **boolean** deleteCustomer(**int** customerId);  **boolean** addToCart(Customer customer, Product product, **int** quantity) **throws** ProductNotFoundException;  **boolean** removeFromCart(Customer customer, Product product);  List<Product> getAllFromCart(Customer customer);  **boolean** placeOrder(Customer customer, List<Map<Product, Integer>> cart, String shippingAddress);  **void** getOrdersByCustomer(Customer customer) **throws** CustomerNotFoundException;    } |

OrderProcessorRepositoryImpl.java

|  |
| --- |
| **package** dao;  **import** entity.\*;  **import** exception.\*;  **import** util.DBConnUtil;  **import** java.sql.\*;  **import** java.sql.Date;  **import** java.util.\*;  **public** **class** OrderProcessorRepositoryImpl **implements** OrderProcessorRepository {    **public** **boolean** createProduct(Product product) {  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement("INSERT INTO products VALUES (?, ?, ?, ?, ?)")) {  stmt.setInt(1, product.getProductId());  stmt.setString(2, product.getProductName());  stmt.setDouble(3, product.getPrice());  stmt.setString(4, product.getDescription());  stmt.setInt(5, product.getQuantity());  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  e.printStackTrace();  **return** **false**;  }  }  **public** **boolean** createCustomer(Customer customer) {  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement("INSERT INTO customers VALUES (?, ?, ?, ?)")  ) {  stmt.setInt(1, customer.getCustomerId());  stmt.setString(2, customer.getCustomerName());  stmt.setString(3, customer.getEmail());  stmt.setString(4, customer.getPassword());  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  **return** **false**;  }  }  **public** **boolean** deleteProduct(**int** productId) {  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement("DELETE FROM products WHERE product\_id = ?")  ) {  stmt.setInt(1, productId);  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  **return** **false**;  }  }  **public** **boolean** deleteCustomer(**int** customerId) {  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement("DELETE FROM customers WHERE customer\_id = ?")  ) {  stmt.setInt(1, customerId);  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  **return** **false**;  }  }    @Override  **public** **boolean** addToCart(Customer customer, Product product, **int** quantity) **throws** ProductNotFoundException {  **try** (Connection conn = DBConnUtil.*getConnection*()) {    PreparedStatement checkStmt = conn.prepareStatement("SELECT \* FROM products WHERE product\_id = ?");  checkStmt.setInt(1, product.getProductId());  ResultSet rs = checkStmt.executeQuery();  **if** (!rs.next()) {  **throw** **new** ProductNotFoundException("Product ID not found: " + product.getProductId());  }    PreparedStatement stmt = conn.prepareStatement("INSERT INTO cart VALUES (NULL, ?, ?, ?)");  stmt.setInt(1, customer.getCustomerId());  stmt.setInt(2, product.getProductId());  stmt.setInt(3, quantity);  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  e.printStackTrace();  **return** **false**;  }  }  **public** **boolean** removeFromCart(Customer customer, Product product) {  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement("DELETE FROM cart WHERE customer\_id = ? AND product\_id = ?")  ) {  stmt.setInt(1, customer.getCustomerId());  stmt.setInt(2, product.getProductId());  **return** stmt.executeUpdate() > 0;  } **catch** (SQLException e) {  **return** **false**;  }  }  **public** List<Product> getAllFromCart(Customer customer) {  List<Product> cartItems = **new** ArrayList<>();  String sql = "SELECT p.product\_id, p.name AS product\_name, p.price, p.description, p.quantity " +  "FROM cart c JOIN products p ON c.product\_id = p.product\_id " +  "WHERE c.customer\_id = ?";  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, customer.getCustomerId());  ResultSet rs = stmt.executeQuery();  **while** (rs.next()) {  Product p = **new** Product();  p.setProductId(rs.getInt("product\_id"));  p.setProductName(rs.getString("product\_name")); // Alias used here  p.setPrice(rs.getDouble("price"));  p.setDescription(rs.getString("description"));  p.setQuantity(rs.getInt("quantity"));  cartItems.add(p);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  **return** cartItems;  }  **public** **boolean** placeOrder(Customer customer, List<Map<Product, Integer>> cart, String shippingAddress) {  Connection conn = **null**;  **try** {  conn = DBConnUtil.*getConnection*();  conn.setAutoCommit(**false**);  PreparedStatement orderStmt = conn.prepareStatement("INSERT INTO orders (customer\_id, order\_date, total\_price, shipping\_address) VALUES (?, CURDATE(), ?, ?)", Statement.***RETURN\_GENERATED\_KEYS***);  **double** total = 0;  **for** (Map<Product, Integer> map : cart) {  **for** (Product p : map.keySet()) {  total += p.getPrice() \* map.get(p);  }  }  orderStmt.setInt(1, customer.getCustomerId());  orderStmt.setDouble(2, total);  orderStmt.setString(3, shippingAddress);  orderStmt.executeUpdate();  ResultSet rs = orderStmt.getGeneratedKeys();  rs.next();  **int** orderId = rs.getInt(1);  PreparedStatement itemStmt = conn.prepareStatement("INSERT INTO order\_items (order\_id, product\_id, quantity) VALUES (?, ?, ?)");  **for** (Map<Product, Integer> map : cart) {  **for** (Product p : map.keySet()) {  itemStmt.setInt(1, orderId);  itemStmt.setInt(2, p.getProductId());  itemStmt.setInt(3, map.get(p));  itemStmt.executeUpdate();  }  }  conn.commit();  **return** **true**;  } **catch** (SQLException e) {  **try** { **if** (conn != **null**) conn.rollback(); } **catch** (SQLException ex) {}  **return** **false**;  } **finally** {  **try** { **if** (conn != **null**) conn.setAutoCommit(**true**); } **catch** (SQLException e) {}  }  }  **public** **void** getOrdersByCustomer(Customer customer) **throws** CustomerNotFoundException {  **boolean** customerExists = **false**;  String sql = "SELECT o.order\_id, o.order\_date, p.name AS product\_name, " +  "oi.quantity, p.price " +  "FROM orders o " +  "JOIN order\_items oi ON o.order\_id = oi.order\_id " +  "JOIN products p ON oi.product\_id = p.product\_id " +  "WHERE o.customer\_id = ?";  **try** (Connection conn = DBConnUtil.*getConnection*();  PreparedStatement stmt = conn.prepareStatement(sql)) {  stmt.setInt(1, customer.getCustomerId());  ResultSet rs = stmt.executeQuery();  **while** (rs.next()) {  customerExists = **true**;  **int** orderId = rs.getInt("order\_id");  String productName = rs.getString("product\_name");  **int** qty = rs.getInt("quantity");  **double** price = rs.getDouble("price");  Date orderDate = rs.getDate("order\_date");  System.***out***.println("Order ID: " + orderId + " | Date: " + orderDate);  System.***out***.println(" - " + productName + " | Qty: " + qty + " | Price: ₹" + price);  }  **if** (!customerExists) {  **throw** **new** CustomerNotFoundException("Customer with ID " + customer.getCustomerId() + " not found or has no orders.");  }  } **catch** (SQLException e) {  e.printStackTrace();  }  }  } |

**Pacakage exception**

CustomerNotFoundException.java

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

OrderNotFoundException.java

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

ProductNotFoundException.java

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

**Package main**

EcomApp.java

|  |
| --- |
| **package** main;  **import** dao.\*;  **import** entity.\*;  **import** exception.\*;  **import** java.util.\*;  **public** **class** EcomApp {  **public** **static** **void** main(String[] args) {  Scanner sc = **new** Scanner(System.***in***);  OrderProcessorRepository repo = **new** OrderProcessorRepositoryImpl();  **while** (**true**) {  System.***out***.println("1. Register Customer");  System.***out***.println("2. Create Product");  System.***out***.println("3. Delete Product");  System.***out***.println("4. Add to cart");  System.***out***.println("5. View cart");  System.***out***.println("6. Place order");  System.***out***.println("7. View Customer Order");  System.***out***.println("8. Exit");  System.***out***.println("Enter your choice: ");  **int** choice = sc.nextInt();  sc.nextLine();  **switch** (choice) {  **case** 1:  System.***out***.print("Enter Customer ID: ");  **int** customerId = sc.nextInt();  sc.nextLine();  System.***out***.print("Enter Customer Name: ");  String name = sc.nextLine();  System.***out***.print("Enter Customer Email: ");  String email = sc.nextLine();  System.***out***.print("Enter Customer Password: ");  String password = sc.nextLine();  System.***out***.print("Enter Customer address: ");  String address = sc.nextLine();  Customer c = **new** Customer(customerId, name, email,address, password);    **if** (repo.createCustomer(c)) {  System.***out***.println("Customer registered successfully!");  } **else** {  System.***out***.println("Customer registration failed.");  } **break**;  **case** 2:  System.***out***.print("Enter Product ID: ");  **int** productId = sc.nextInt();  sc.nextLine();  System.***out***.print("Enter Product Name: ");  String productName = sc.nextLine();  System.***out***.print("Enter Product Price: ");  **double** price = sc.nextDouble();  sc.nextLine();  System.***out***.print("Enter Product Description: ");  String description = sc.nextLine();  System.***out***.print("Enter Stock Quantity: ");  **int** stock = sc.nextInt();  Product p = **new** Product(productId, productName, price, description,stock);  **if** (repo.createProduct(p)) {  System.***out***.println("Product created successfully!");  } **else** {  System.***out***.println("Failed to create product.");  }  **break**;  **case** 3:  System.***out***.print("Enter Product ID to delete: ");  **int** deleteId = sc.nextInt();  **if** (repo.deleteProduct(deleteId)) {  System.***out***.println("Product deleted successfully!");  } **else** {  System.***out***.println("Failed to delete product.");  }  **break**;  **case** 4:  **try** {  System.***out***.print("Enter Customer ID: ");  **int** addCustomerId = sc.nextInt();  System.***out***.print("Enter Product ID: ");  **int** addProductId = sc.nextInt();  System.***out***.print("Enter Quantity: ");  **int** addQty = sc.nextInt();  Customer ac = **new** Customer(); ac.setCustomerId(addCustomerId);  Product ap = **new** Product(); ap.setProductId(addProductId);  **if**(repo.addToCart(ac, ap, addQty))  {  System.***out***.println("Product added to cart successfully!");  }  **else**  {  System.***out***.println("Product not added to cart successfully!");  }  } **catch** (exception.ProductNotFoundException e) {  System.***out***.println(e.getMessage());  }  **break**;  **case** 5:  System.***out***.print("Enter Customer ID: ");  **int** viewCustomerId = sc.nextInt();  Customer vc = **new** Customer();  vc.setCustomerId(viewCustomerId);  List<Product> cartItems = repo.getAllFromCart(vc);  **if** (cartItems.isEmpty()) {  System.***out***.println("Your cart is empty or customer ID is invalid.");  } **else** {  System.***out***.println("Items in your cart:");  **for** (Product prod : cartItems) {  System.***out***.println(prod.getProductName() + " - ₹" + prod.getPrice());  }  }  **break**;  **case** 6:  System.***out***.print("Enter Customer ID: ");  **int** orderCustomerId = sc.nextInt();  sc.nextLine();  System.***out***.print("Enter Shipping Address: ");  String shippingAddress = sc.nextLine();  Customer oc = **new** Customer();  oc.setCustomerId(orderCustomerId);  List<Product> items = repo.getAllFromCart(oc);  List<Map<Product, Integer>> orderCart = **new** ArrayList<>();  **for** (Product prod : items) {  Map<Product, Integer> map = **new** HashMap<>();  map.put(prod, 1);  orderCart.add(map);  }  **if**(repo.placeOrder(oc, orderCart, shippingAddress))  {  System.***out***.println("order placed successfully!");  }  **else**  {  System.***out***.println("order not placed successfully!");  }  **break**;  **case** 7:  **try** {  System.***out***.print("Enter Customer ID: ");  **int** custId = sc.nextInt();  Customer customer = **new** Customer();  customer.setCustomerId(custId);  repo.getOrdersByCustomer(customer);  } **catch** (CustomerNotFoundException e) {  System.***out***.println(e.getMessage());  } **catch** (Exception e) {  System.***out***.println("An unexpected error occurred: " + e.getMessage());  }  **break**;  **case** 8:  System.*exit*(0);  sc.close();  }  }      }  } |

**Package util**

DBConnUtil.java

|  |
| --- |
| **package** util;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.SQLException;  **public** **class** DBConnUtil {  **public** **static** Connection getConnection() {  **try** {  String connStr = DBPropertyUtil.*getPropertyString*("C:\\Users\\ELCOT\\Workspace\\Ecom\\src\\db.properties");  **return** DriverManager.*getConnection*(connStr);  } **catch** (SQLException e) {  **return** **null**;  }  }  } |

DBPropertyUtil.java

|  |
| --- |
| **package** util;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.util.Properties;  **public** **class** DBPropertyUtil {  **public** **static** String getPropertyString(String filename) {  Properties props = **new** Properties();  **try** (FileInputStream input = **new** FileInputStream(filename)) {  props.load(input);  **return** "jdbc:mysql://" + props.getProperty("host") + ":" + props.getProperty("port") + "/" + props.getProperty("dbname") +  "?user=" + props.getProperty("username") + "&password=" + props.getProperty("password");  } **catch** (IOException e) {  **return** **null**;  }  }  } |

Db.properties

|  |
| --- |
| host=localhost  port=3306  dbname=ECOM  username=root  password=@Tamil9988 |

**Package test**

EcomAppTest.java

|  |
| --- |
| **package** test;  **import** java.util.\*;  **import** dao.OrderProcessorRepository;  **import** dao.OrderProcessorRepositoryImpl;  **import** entity.\*;  **import** exception.\*;  **import** org.junit.jupiter.api.Test;  **import** **static** org.junit.jupiter.api.Assertions.\*;  **public** **class** EcomAppTest {  OrderProcessorRepository repo = **new** OrderProcessorRepositoryImpl();  @Test  **public** **void** testCreateProductSuccess() {  Product product = **new** Product(134, "smartwatch", 133.99, "wearables",30);  **boolean** result = repo.createProduct(product);  *assertTrue*(result);  }  @Test  **public** **void** testAddToCartSuccess() **throws** ProductNotFoundException {  Customer customer = **new** Customer(1, "Test User", "test@example.com","chennai", "test123");  Product product = **new** Product(102, "Test Cart Product", 50.0, "Cart test product", 10);  repo.createCustomer(customer);  repo.createProduct(product);  **boolean** result = repo.addToCart(customer, product, 2);  *assertTrue*(result);  }  @Test  **public** **void** testPlaceOrderSuccess() **throws** ProductNotFoundException {  Customer customer = **new** Customer(2, "Order User", "order@example.com","pune", "pass123");  Product product = **new** Product(103, "OrderProduct", 120.0, "Test Order", 10);  repo.createCustomer(customer);  repo.createProduct(product);  repo.addToCart(customer, product, 1);  List<Map<Product, Integer>> cart = **new** ArrayList<>();  Map<Product, Integer> item = **new** HashMap<>();  item.put(product, 1);  cart.add(item);  **boolean** result = repo.placeOrder(customer, cart, "123 Order Lane");  *assertTrue*(result);  }  @Test  **public** **void** testCustomerNotFoundException() {  **int** invalidCustomerId = 99999;  Customer customer = **new** Customer();  customer.setCustomerId(invalidCustomerId);  Exception exception = *assertThrows*(CustomerNotFoundException.**class**, () -> {  repo.getOrdersByCustomer(customer);  });  *assertEquals*("Customer with ID " + invalidCustomerId + " not found or has no orders.", exception.getMessage());  }  @Test  **public** **void** testProductNotFoundException() {  Customer customer = **new** Customer(1, "Fake", "fake@x.com", "mumbai","123");  Product invalidProduct = **new** Product();  invalidProduct.setProductId(9999);  Exception exception = *assertThrows*(ProductNotFoundException.**class**, () -> {  repo.addToCart(customer, invalidProduct, 1);  });  *assertEquals*("Product ID not found: 9999", exception.getMessage());  }  } |