CASE STUDY :E COMMERCE

SUBMITTED BY:SRINIDHI.V

Create following tables in SQL Schema with appropriate class and write the unit test case for the Ecommerce application.
Schema Design:
1. customers table:
• customer_id (Primary Key)
• name
• email
• password

```
CREATE DATABASE IF NOT EXISTS ecomm;
USE ecomm;

-- 1. Customers Table
CREATE TABLE IF NOT EXISTS customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL
);
```

2. products table:
• product_id (Primary Key)
• name
• price
• description
• stockQuantity

```
-- 2. Products Table
CREATE TABLE IF NOT EXISTS products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    description TEXT,
    stockQuantity INT NOT NULL
);
```

3. cart table:
• cart_id (Primary Key)
• customer_id (Foreign Key)
• product_id (Foreign Key)
• quantity

```
-- 3. Cart Table
CREATE TABLE IF NOT EXISTS cart (
    cart_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    quantity INT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE
CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE
CASCADE
);
```

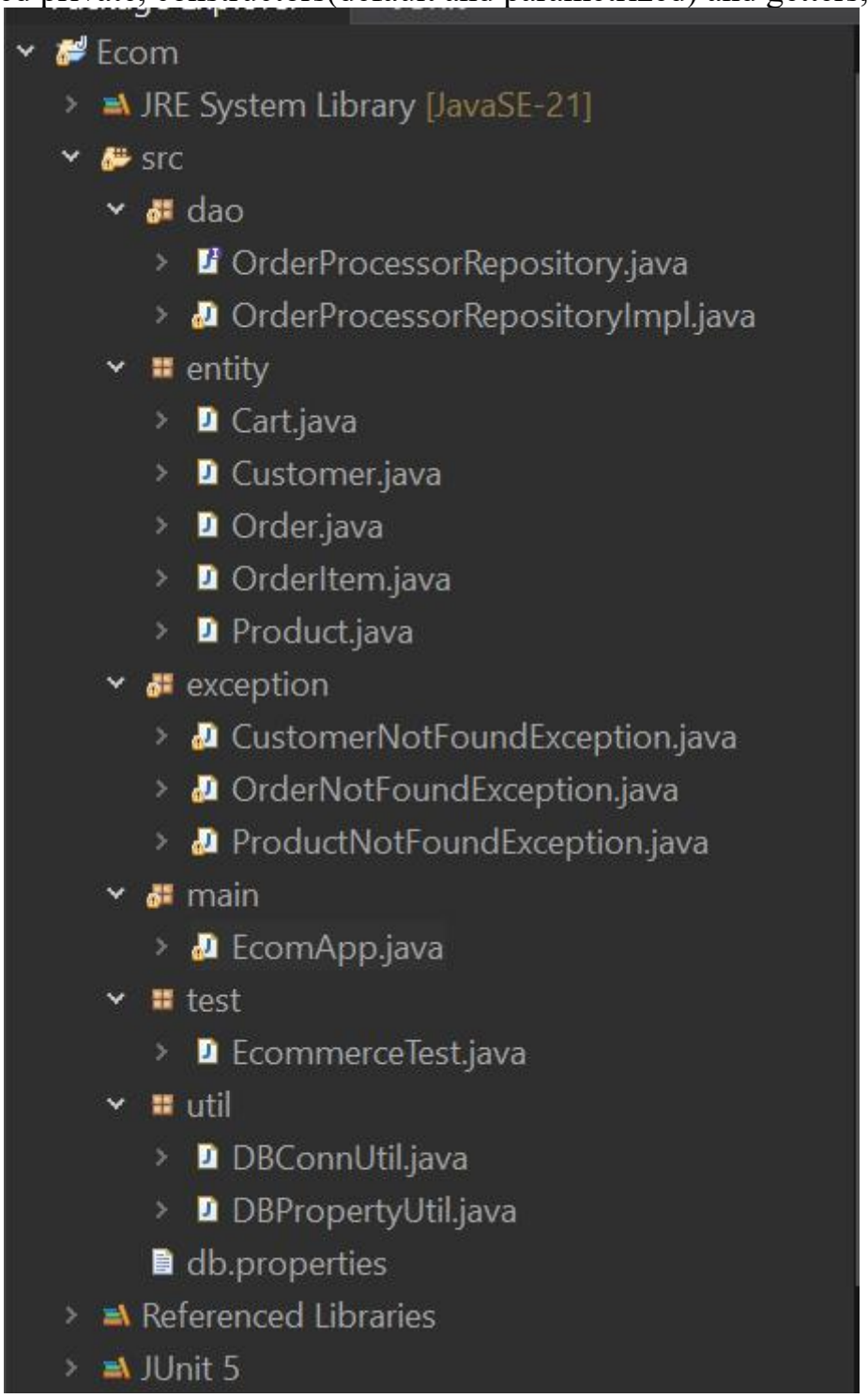
4. orders table:
- **order_id (Primary Key)**
- **customer_id (Foreign Key)**
- **order_date**
- **total_price**
- **shipping_address**

```
CREATE TABLE IF NOT EXISTS orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    total_price DECIMAL(10,2),
    shipping_address TEXT,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE
CASCADE
);
```

5. order_items table (to store order details):
- **order_item_id (Primary Key)**
- **order_id (Foreign Key)**
- **product_id (Foreign Key)**
- **quantity**

```
- 5. Order Items Table
CREATE TABLE IF NOT EXISTS order_items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE
CASCADE
);
```

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)



6. Service Provider Interface/Abstract class:

Keep the interfaces and implementation classes in package dao

• Define an OrderProcessorRepository interface/abstract class with methods for adding/removing products to/from the cart and placing orders. The following methods will interact with database.

1. createProduct()

parameter: Product product

return type: boolean

2. createCustomer()
parameter: Customer customer
return type: boolean
3. deleteProduct()
parameter: productId
return type: boolean
4. deleteCustomer(customerId)
parameter: customerId
return type: boolean
5. addToCart(): insert the product in cart.
parameter: Customer customer, Product product, int quantity
return type: boolean
6. removeFromCart(): delete the product in cart.
parameter: Customer customer, Product product
return type: boolean
7. getAllFromCart(Customer customer): list the product in cart for a customer.
parameter: Customer customer
return type: list of product
8. placeOrder(Customer customer, List<Map<Product,quantity>>, string shippingAddress): should update order table and orderItems table.
1. parameter: Customer customer, list of product and quantity
2. return type: boolean
9. getOrdersByCustomer()
1. parameter: customerid
2. return type: list of product and quantity

ORDER PROCESSOR REPOSITORY

Customer.java | Product.java | DBConnUtil.java | OrderProcess... | EcommerceTes...

```java
package dao;

import entity.Customer;
import entity.Product;
import java.util.List;
import java.util.Map;

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);
    boolean removeFromCart(Customer customer, Product product);
    List<Product> getAllFromCart(Customer customer);
    boolean placeOrder(Customer customer, List<Map<Product, Integer>> products, String shi
    List<Map<Product, Integer>> getOrdersByCustomer(int customerId);
}
```

7. Implement the above interface in a class called **OrderProcessorRepositoryImpl in package dao**.

```java
package dao;

import entity.Customer;
import entity.Product;
import exception.CustomerNotFoundException;
import exception.OrderNotFoundException;
import exception.ProductNotFoundException;
import util.DBConnUtil;

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

public class OrderProcessorRepositoryImpl implements OrderProcessorRepository {

    private Connection conn;

    public OrderProcessorRepositoryImpl() {
        conn = DBConnUtil.getConnection();
    }

    @Override
    public boolean createCustomer(Customer customer) {
        String query = "INSERT INTO customers (name, email, password) VALUES (?, ?, ?)";
        try (PreparedStatement ps = conn.prepareStatement(query)) {
            ps.setString(1, customer.getName());
            ps.setString(2, customer.getEmail());
            ps.setString(3, customer.getPassword());
            return ps.executeUpdate() > 0;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
```

p\JAAVAAA\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-1957\jre\bin\javaw.exe (Apr 20, 2025, 3

Connect your application to the SQL database:

8. Write code to establish a connection to your SQL database.

• 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 file containing connection details like hostname, dbname, username, password, port number and returns a connection string.

DBCONNUTIL

Customer.java | Product.java | DBConnUtil.java | OrderProcess... | OrderProcess...

```java
package util;

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

public class DBConnUtil {

    private static Connection conn;

    public static Connection getConnection() {
        if (conn == null) {
            try {
                String url = DBPropertyUtil.getPropertyString("db.properties");
                System.out.println("Connection URL: " + url); // Debug line
                if (url != null) {
                    conn = DriverManager.getConnection(url);
                } else {
                    System.out.println("Failed to load DB connection string.");
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return conn;
    }
}
```

DBPROPERTYUTIL

```java
package util;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class DBPropertyUtil {

    public static String getPropertyString(String filename) {
        Properties props = new Properties();
        try (InputStream input = DBPropertyUtil.class.getClassLoader().getResourceAsStream
            if (input == null) {
                System.out.println("Sorry, unable to find " + filename);
                return null;
            }
            props.load(input);
        } catch (IOException ex) {
            ex.printStackTrace();
            return null;
        }

        String host = props.getProperty("host");
        String port = props.getProperty("port");
        String dbname = props.getProperty("dbname");
        String username = props.getProperty("username");
        String password = props.getProperty("password");

        return "jdbc:mysql://" + host + ":" + port + "/" + dbname +
                "?user=" + username + "&password=" + password +
                "&useSSL=false&serverTimezone=UTC";
    }
}
```

9. Create the exceptions in package myexceptions and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

• CustomerNotFoundException: throw this exception when user enters an invalid customer id which doesn't exist in db

• ProductNotFoundException: throw this exception when user enters an invalid product

id which doesn't exist in db

• OrderNotFoundException: throw this exception when user enters an invalid order id which doesn't exist in db

CUSTOMERNOTDOUNDEXCEPTION

```java
package exception;

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

PRODUCTNOTFOUNDEXCEPTION

```java
package exception;

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

ORDERNOTFOUNDEXCEPTION

```java
package exception;

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

10. Create class named EcomApp with main method in app Trigger all the methods in service
implementation class by user choose operation from the following menu.
1. Register Customer.
2. Create Product.
3. Delete Product.
4. Add to cart.
5. View cart.
6. Place order.
7. View Customer Order

```
EcomApp [Java Application] C:\Users\srini\OneDrive\Desktop\JAAVAAA\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-19
Connection URL: jdbc:mysql://localhost:3306/ecomm?user=root&password=Mickey12@&useSSL=false&serverTimezone=UTC

==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option:
```

1. REGISTER CUSTOMER

```
Choose option: 1
Enter Name:
BUJJI
Enter Email:
BUJJI@123
Enter Password:
BUJJI12
Customer registered successfully.
```

11	BUJJI	BUJJI@123	BUJJI12
NULL	NULL	NULL	NULL

2.CREATE A PRODUCT

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 2
Enter Product Name:
BLUETOOTH MOUSE
Enter Price:
500
Enter Description:
WIRELESS BLUETOOTH MOUSE
Enter Stock Quantity:
2
Product created successfully.
```

3.DELETE PRODUCT

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 3
Enter Product ID to delete:
41
Product deleted.
```

4. ADD TO CART

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 4
Enter Customer ID:
1
Enter Product ID:
1
Enter Quantity:
1
Product added to cart.
```

5. VIEW CART

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 5
Enter Customer ID:
1
Cart contents:
1: iphone
```

6. PLACE ORDER

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 6
Enter Customer ID:
1
Enter Shipping Address:
CHENNAI INDIA
Enter quantity for iphone:
2
Order placed successfully.
```

7. VIEW CUSTOMERS ORDERS

```
==== Ecommerce Application Menu ====
1. Register Customer
2. Create Product
3. Delete Product
4. Add to Cart
5. View Cart
6. Place Order
7. View Customer Orders
8. Exit
Choose option: 7
Enter Customer ID:
1
Customer Orders:
iphone x 1
iphone x 2
```

SQL ORDERS

```
SELECT * FROM orders;
```

order_id	customer_id	order_date	total_price	shipping_address
1	1	2025-04-20 15:07:29	45000.00	chennai tn
2	1	2025-04-20 16:24:28	90000.00	CHENNAI INDIA
NULL	NULL	NULL	NULL	NULL

SQL PRODUCTS

```
SELECT * FROM PRODUCTS;
```

product_id	name	price	description	stockQuantity
1	iphone	45000.00	new iphone 11	2
3	Smartwatch	8999.00	Fitness focused smartwatch with heart rate mo...	80
4	Bluetooth Earbuds	5999.00	Wireless noise canceling earbuds with charging ...	100
5	Smart TV	29999.00	50-inch 4K Ultra HD Smart LED TV with built in a...	40
6	Air Conditioner (AC)	35999.00	1.5 Ton Split AC with inverter technology	30
7	Laptop	65999.00	15.6 inch laptop with Intel i5 processor and 16G...	60
8	iPhone	79999.00	Apple iPhone 14 with 128GB storage	45
9	iPad	44999.00	Apple iPad 10.9-inch with A14 Bionic chip	50
10	Tablet	18999.00	10-inch Android tablet with expandable storage	70
42	Power Adapter	999.00	Fast-charging USB C wall adapter (20W)	200
43	BLUETOOTH MOUSE	500.00	WIRELESS BLUETOOTH MOUSE	2
NULL	NULL	NULL	NULL	NULL

SQL ORDER ITEMS

```
SELECT * FROM order_items;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_item_id	order_id	product_id	quantity
1	1	1	1
2	2	1	2
NULL	NULL	NULL	NULL

SQL CUSTOMERS

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_id	name	email	password
1	sri	sri@12	sri
2	Fahimunisha	fahimunisha@gmail.com	pass@123
3	Shagir	shagir@gmail.com	pass@123
4	Sajith	sajith@gmail.com	pass@123
5	Mickey	mickey@gmail.com	pass@123
6	Nidhi	nidhi@gmail.com	pass@123
7	Jokiee	jokiee@gmail.com	pass@123
8	Jack	jack@gmail.com	pass@123
9	Ram	ram@gmail.com	pass@123
10	Sangee	sangee@gmail.com	pass@123
11	BUJJI	BUJJI@123	BUJJI12
NULL	NULL	NULL	NULL