

Nama / NIM : Salmiah Afifah / 2411523005

Jurusan : Sistem Informasi

Mata Kuliah : PBO

1. Pada sebuah toko kain mengalami kesulitan dalam mengelola stok kain (jenis, warna, panjang per meter), pencatatan transaksi penjualan masih manual → rawan salah hitung, sulit melacak pelanggan tetap dan riwayat pembelian dan tidak ada sistem laporan penjualan harian/mingguan untuk pemilik toko.

**Solusi:**

Dengan membangun **Sistem Informasi Manajemen Penjualan Kain** berbasis PBO dengan fitur:

- **Manajemen Produk:** tambah, ubah, hapus data kain (jenis, warna, harga per meter).
- **Transaksi Penjualan:** input pembelian pelanggan, otomatis menghitung total harga.
- **Manajemen Pelanggan:** menyimpan data pelanggan tetap dan riwayat pembelian.
- **Laporan Penjualan:** menampilkan ringkasan transaksi harian/mingguan.

**Pendekatan PBO:**

- **Encapsulation:** Data kain dan pelanggan disimpan dalam objek dengan akses terbatas.
- **Inheritance:** Kelas Product sebagai superclass, diturunkan menjadi Fabric (kain), Accessory (aksesoris).
- **Polymorphism:** Metode calculatePrice() bisa berbeda untuk kain (per meter) dan aksesoris (per item).
- **Abstraction:** Interface CRUDOperations untuk operasi database (create, read, update, delete).

2.

```
// a. Class, Object, Constructor
class Product {
    protected String name;
    protected double price;

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
}
```

```

    }

    public double calculatePrice(int quantity) {
        return price * quantity;
    }
}

// c. Inheritance (superclass & subclass)
class Fabric extends Product {
    private double lengthPerMeter;

    public Fabric(String name, double price, double lengthPerMeter) {
        super(name, price);
        this.lengthPerMeter = lengthPerMeter;
    }

    @Override
    public double calculatePrice(int meters) {
        return price * meters;
    }
}

```

```

class Accessory extends Product {

```

```

    public Accessory(String name, double price) {
        super(name, price);
    }
}

```

```

// b. Interface & Implementasi

```

```
interface CRUDOperations {
```

```
    void create();
```

```
    void read();
```

```
    void update();
```

```
    void delete();
```

```
}
```

```
class DatabaseHandler implements CRUDOperations {
```

```
    @Override
```

```
        public void create() { System.out.println("Insert data to DB"); }
```

```
    @Override
```

```
        public void read() { System.out.println("Read data from DB"); }
```

```
    @Override
```

```
        public void update() { System.out.println("Update data in DB"); }
```

```
    @Override
```

```
        public void delete() { System.out.println("Delete data from DB"); }
```

```
}
```

```
// d. Perulangan, Percabangan, Perhitungan
```

```
class Transaction {
```

```
    private Product product;
```

```
    private int quantity;
```

```
    public Transaction(Product product, int quantity) {
```

```
        this.product = product;
```

```
        this.quantity = quantity;
```

```
    }
```

```
    public void processTransaction() {
```

```

        double total = product.calculatePrice(quantity);
        if (total > 500000) {
            System.out.println("Diskon 10% diterapkan!");
            total *= 0.9;
        }
        System.out.println("Total harga: Rp " + total);
    }
}

// e. Manipulasi String & Date
import java.text.SimpleDateFormat;
import java.util.Date;

class Utility {
    public static void printReceipt(String customerName) {
        String upperName = customerName.toUpperCase();
        String date = new SimpleDateFormat("dd/MM/yyyy HH:mm").format(new Date());
        System.out.println("Receipt for: " + upperName);
        System.out.println("Date: " + date);
    }
}

// f. Exception Handling
class ExceptionDemo {
    public static void divide(int a, int b) {
        try {
            int result = a / b;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {

```

```
System.out.println("Error: Division by zero!");
```

```
}
```

```
}  
}
```

```
// g. Collection Framework
```

```
import java.util.ArrayList;
```

```
class Inventory {
```

```
    private ArrayList<Product> products = new ArrayList<>();
```

```
    public void addProduct(Product p) {
```

```
        products.add(p)
```

```
    public void showProducts() {
```

```
        for (Product p : products) {
```

```
            System.out.println(p.name + " - Rp " + p.price);
```

```
        }
```

```
    }
```

```
}
```

```
// h. JDBC & CRUD (contoh sederhana)
```

```
import java.sql.*;
```

```
class FabricDB implements CRUDOperations {
```

```
    private Connection conn;
```

```
public FabricDB() {  
    try {  
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/tokokain",  
"root", "");  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}  
  
@Override  
public void create() {  
    try {  
        Statement stmt = conn.createStatement();  
        stmt.executeUpdate("INSERT INTO fabric(name, price) VALUES('Katun',  
50000)");  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}  
  
@Override  
public void read() {  
    try {  
        Statement stmt = conn.createStatement();  
        ResultSet rs = stmt.executeQuery("SELECT * FROM fabric");  
        while (rs.next()) {  
            System.out.println(rs.getString("name") + " - Rp " + rs.getDouble("price"));  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```

```

    }
}

@Override
public void update() {
    try {
        Statement stmt = conn.createStatement();
        stmt.executeUpdate("UPDATE fabric SET price=60000 WHERE name='Katun'");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Override
public void delete() {
    try {
        Statement stmt = conn.createStatement();
        stmt.executeUpdate("DELETE FROM fabric WHERE name='Katun'");
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

// Main Program
public class Main {
    public static void main(String[] args) {
        Fabric kain = new Fabric("Katun", 50000, 1.0);
        Transaction trx = new Transaction(kain, 15);
    }
}

```

```
trx.processTransaction();
```

```
Utility.printReceipt("Salmiah");
```

```
ExceptionDemo.divide(10, 0);
```

```
Inventory inv = new Inventory();
```

```
inv.addProduct(kain);
```

```
inv.addProduct(new Accessory("Benang", 10000));
```

```
inv.showProducts();
```

```
FabricDB db = new FabricDB();
```

```
db.create();
```

```
db.read();
```

```
db.update();
```

```
db.delete();
```

```
}
```

```
}
```