Toko Barang

Anda menerima sebuah proyek sistem aplikasi toko barang yang mengelola inventaris dan penjualan barang. Proyek ini sudah ada kode program awal. Dan setelah anda melihat, kode program tersebut melanggar prinsip-prinsip SOLID.

Refaktor kode program berikut untuk memperbaiki pelanggaran prinsip SOLID dengan mengikuti prinsip Single Responsibility Principle (SRP), Open-Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP), dan Dependency Inversion Principle (DIP).

Catatan: ubah logik apabila diperlukan untuk memenuhi SOLID. Jika mengikuti kunci jawaban, perubahan logik harusnya sangat minim (hanya di beberapa fungsi), yang banyak adalah perubahan struktur (kelas, fungsi, *interface* dan sejenisnya).

Yang bisa kalian lakukan:

1. Menambah kelas/*interface*/method/*abstract class*
2. Mengubah kepemilikan method
3. Merename method
4. Dan lain-lain.

```
public class Product {
    private String name;
    private double price;
    private static List<Product> shoppingCarts = new ArrayList<>();

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

    public String getName() {
        return name;
    }
```

```
    public double getPrice() {
        return price;
    }

    public static void addProductToShoppingCart(Product product) {
        shoppingCarts.add(product);
    }

    public static void removeProductFromShoppingCart(Product product) {
        shoppingCarts.remove(product);
    }

    public static double calculateShoppingCartsTotalPrice() {
        double total = 0;
        for (Product product : shoppingCarts) {
            total += product.getPrice();
        }
        return total;
    }

    // Metode untuk menghitung pajak
    public double calculateTax() {
        return price * 0.1;
    }
}

public class FoodProduct extends Product {
    public FoodProduct(String name, double price) {
        super(name, price);
    }

    // Override metode calculateTax()
    @Override
    public double calculateTax() {
        return 0; // Tidak ada pajak untuk produk makanan
    }
}

public class ElectronicsProduct extends Product {
    public ElectronicsProduct(String name, double price) {
```

```
        super(name, price);
    }

    // Override metode calculateTax()
    @Override
    public double calculateTax() {
        return price * 0.2; // Pajak khusus untuk produk elektronik
    }
}

public class PaymentProcessor {
    public void processPaymentWithDebitCard(double amount) {
        // Process payment logic
        System.out.println("Payment processed successfully with debit card: $" +
amount);
    }

    public void processPaymentWithCreditCard(double amount) {
        // Process payment logic
        System.out.println("Payment processed successfully with credit card: $" +
amount);
    }

    public void processPaymentWithCash(double amount) {
        // Process payment logic
        System.out.println("Payment processed successfully with cash: $" + amount);
    }
}

public class Order {
    private ShoppingCart shoppingCart;
    private PaymentProcessor paymentProcessor;

    public Order(ShoppingCart shoppingCart) {
        this.shoppingCart = shoppingCart;
        this.paymentProcessor = new PaymentProcessor();
    }

    public void checkout(String checkoutMethod) {
        double totalPrice = shoppingCart.calculateTotalPrice();
        if (checkoutMethod.equalsIgnoreCase("Cash")) {
```

```
            paymentProcessor.processPaymentWithCash(totalPrice);
        } else if (checkoutMethod.equalsIgnoreCase("CreditCard")) {
            paymentProcessor.processPaymentWithCreditCard(totalPrice);
        } else if (checkoutMethod.equalsIgnoreCase("DebitCard")) {
            paymentProcessor.processPaymentWithDebitCard(totalPrice);
        } else {
            throw new IllegalArgumentException("Invalid payment method.");
        }
    }
}
```

Kumpulkan:

1. SOLID.txt yang berisikan penjelasan setiap aspek S, O, L, I dan D pada kode yang kalian refaktor.

   Contoh:
   S: Fungsi X dan Y dipindah dari kelas ABC ke kelas baru DEF agar konsep Single Responsibility terpenuhi. If else … dipindahkan
   O: Dibuatkan … agar konsep Open-closed principle terpenuhi sehingga kedepannya …, karena if else … dipindahkan menjadi … maka … dan kedepannya ….
   dst.
2. Kumpulkan SOLID.txt dan kode yang telah kalian refaktor pada zip bernama **SOLID_NIM.zip**