

# Bab 10

## Prinsip SOLID dalam Desain OOP

### 10.1 SRP, OCP, dan LSP

#### Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.1: Menganalisis code smell dan melakukan refactoring

#### Konsep Penting

**SOLID** adalah lima prinsip desain yang membuat perangkat lunak lebih mudah dipahami, fleksibel, dan maintainable [1].

#### 10.1.1 Single Responsibility Principle (SRP)

Satu class sebaiknya memiliki satu alasan untuk berubah.

#### 10.1.2 Open/Closed Principle (OCP)

Class harus terbuka untuk extension tetapi tertutup untuk modification.

#### 10.1.3 Liskov Substitution Principle (LSP)

Objek subclass harus dapat menggantikan objek superclass tanpa merusak perilaku program.

#### Catatan

Jika class sering berubah karena banyak alasan, itu adalah tanda pelanggaran SRP.

## 10.2 ISP, DIP, dan Contoh Refactoring

### 10.2.1 Interface Segregation Principle (ISP)

Gunakan interface kecil dan spesifik daripada interface besar yang memaksa class mengimplementasikan method yang tidak dibutuhkan.

### 10.2.2 Dependency Inversion Principle (DIP)

High-level module tidak boleh bergantung pada low-level module. Keduanya harus bergantung pada abstraksi.

```
1 interface Notifier {  
2     void kirim(String pesan);  
3 }  
4  
5 class EmailNotifier implements Notifier {  
6     public void kirim(String pesan) {  
7         System.out.println("Email: " + pesan);  
8     }  
9 }  
10  
11 class LayananNotifikasi {  
12     private Notifier notifier;  
13  
14     public LayananNotifikasi(Notifier notifier) {  
15         this.notifier = notifier;  
16     }  
17  
18     public void kirimPromo() {  
19         notifier.kirim("Promo baru!");  
20     }  
21 }
```

Kode Program 10.1: Contoh DIP dengan Interface

### Aktivitas Pembelajaran

1. **Identifikasi Code Smell:** Temukan pelanggaran SRP pada class layanan.
2. **Refactoring:** Ubah class yang terlalu besar menjadi beberapa class kecil.
3. **Diskusi:** Contoh pelanggaran LSP pada hierarki class.
4. **Praktik:** Terapkan DIP pada modul pengiriman notifikasi.

## Latihan dan Refleksi

1. Jelaskan masing-masing prinsip SOLID dengan contoh singkat.
2. Berikan contoh pelanggaran OCP dan cara memperbaikinya.
3. Mengapa ISP penting saat mendesain API?
4. Buat desain sederhana yang menerapkan DIP untuk layanan pembayaran.
5. **Refleksi:** Prinsip SOLID mana yang paling sulit Anda terapkan?

## Asesmen (Evaluasi Kinerja)

### Instrumen Penilaian untuk Sub-CPMK 4.1

#### A. Pilihan Ganda

1. Prinsip yang menyatakan class memiliki satu alasan untuk berubah adalah:
  - (a) SRP
  - (b) OCP
  - (c) LSP
  - (d) DIP
2. DIP menekankan ketergantungan pada:
  - (a) Implementasi konkret
  - (b) Abstraksi
  - (c) Class final
  - (d) Static method

#### B. Tugas Praktik

- Refactor class **OrderService** agar mengikuti SRP dan OCP.

**Rubrik Penilaian:** Lihat Lampiran A

## Checklist Pencapaian Kompetensi

*Centang item berikut setelah Anda yakin telah menguasainya:*

☐ Saya memahami prinsip SRP, OCP, dan LSP

- ☐ Saya memahami prinsip ISP dan DIP
- ☐ Saya dapat mengidentifikasi pelanggaran SOLID
- ☐ Saya dapat melakukan refactoring dasar sesuai SOLID
- ☐ Saya dapat menerapkan SOLID pada desain sederhana

### Rangkuman

Prinsip SOLID membantu mengurangi ketergantungan, meningkatkan modularitas, dan memudahkan perubahan, sebagaimana dijelaskan dalam konsep arsitektur bersih [1]. Dengan menerapkannya, kode menjadi lebih bersih dan tahan terhadap perubahan.

**Kata Kunci:** *SOLID, SRP, OCP, LSP, ISP, DIP*

# Daftar Pustaka

- [1] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.