Bab 6

Polymorphism dan Dynamic Binding

6.1 Polymorphism dan Method Overloading

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 3.2: Menerapkan polymorphism dalam desain aplikasi

Konsep Penting

Polymorphism adalah kemampuan suatu object untuk mengambil berbagai bentuk [1]. Dalam Java, polymorphism muncul dalam dua bentuk: compile-time (overloading) dan runtime (overriding).

6.1.1 Compile-time Polymorphism (Overloading)

Method overloading terjadi ketika terdapat beberapa method dengan nama sama, tetapi parameter berbeda.

```
class Kalkulator {
    public int tambah(int a, int b) {
        return a + b;
    }

    public double tambah(double a, double b) {
        return a + b;
    }

    public int tambah(int a, int b, int c) {
        return a + b + c;
    }
}
```

Kode Program 6.1: Contoh Overloading

6.1.2 Keuntungan Overloading

- Memudahkan penggunaan API karena nama method konsisten
- Mengurangi kompleksitas nama method
- Membuat kode lebih mudah dibaca

6.1.3 Perbandingan Overloading vs Overriding

Aspek	Overloading	Overriding
Binding Time	Compile-time (Static)	Runtime (Dynamic)
Class	Dalam class yang sama	Antara superclass dan subclass
Keyword	Tidak ada	`@Override` (Opsional tapi disarankan)
Parameter	Harus berbeda	Harus sama
Return Type	Boleh berbeda	Harus sama atau sub-type

Tabel 6.1: Perbandingan Overloading dan Overriding

6.2 Runtime Polymorphism dan Dynamic Dispatch

6.2.1 Overriding dan Dynamic Binding

Runtime polymorphism terjadi saat method yang dipanggil ditentukan pada saat runtime berdasarkan objek aktual.

```
1  class Bentuk {
2      public double luas() {
3          return 0.0;
4      }
5  }
6
7  class Lingkaran extends Bentuk {
8      private double r;
9
10     public Lingkaran(double r) {
11         this.r = r;
12     }
13
14     @Override
15     public double luas() {
```

```
16          return Math.PI * r * r;
17      }
18  }
19
20  class Persegi extends Bentuk {
21      private double s;
22
23      public Persegi(double s) {
24          this.s = s;
25      }
26
27      @Override
28      public double luas() {
29          return s * s;
30      }
31  }
32
33  public class Demo {
34      public static void main(String[] args) {
35          Bentuk b1 = new Lingkaran(7);
36          Bentuk b2 = new Persegi(4);
37          System.out.println(b1.luas());
38          System.out.println(b2.luas());
39      }
40  }
```

Kode Program 6.2: Runtime Polymorphism

6.2.2 Polymorphism pada Interface

Polymorphism juga terjadi ketika menggunakan reference bertipe interface.

Aktivitas Pembelajaran

1. **Eksperimen Overloading**: Buat class **`Printer`** dengan method `cetak()` untuk tipe data berbeda.

2. **Dynamic Dispatch**: Buat array bertipe superclass yang berisi beberapa subclass dan panggil method yang dioverride.

3. **Studi Kasus**: Implementasikan interface **`Pembayaran`** dengan beberapa jenis pembayaran.

4. **Diskusi**: Jelaskan manfaat polymorphism dalam pengembangan framework.

Latihan dan Refleksi

1. Apa perbedaan overloading dan overriding?
2. Buat class **`Hewan`** dengan method `suara()`. Buat subclass **`Anjing`** dan **`Kucing`** yang mengoverride method tersebut.
3. Buat contoh penggunaan interface yang menunjukkan polymorphism.
4. Mengapa polymorphism penting untuk extensibility aplikasi?
5. **Refleksi**: Dalam proyek Anda, bagian mana yang paling terbantu oleh polymorphism?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 3.2

A. Pilihan Ganda

1. Runtime polymorphism terjadi karena:
   (a) Overloading
   (b) Overriding
   (c) Constructor
   (d) Static method
2. Method yang dipanggil pada runtime ditentukan oleh:
   (a) Tipe reference
   (b) Tipe objek aktual
   (c) Nama method
   (d) Jumlah parameter

B. Tugas Praktik

- Buat hierarki class **`Notifikasi`** dengan subclass **`Email`** dan **`SMS`**. Demonstrasikan polymorphism saat mengirim notifikasi.

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya memahami konsep polymorphism

☐ Saya dapat membuat method overloading

☐ Saya dapat menerapkan method overriding

☐ Saya memahami dynamic dispatch pada runtime

☐ Saya dapat menggunakan polymorphism dengan interface

Rangkuman

Polymorphism memungkinkan satu interface digunakan untuk banyak bentuk objek [1].

Overloading terjadi di compile-time, sedangkan overriding terjadi di runtime melalui dynamic dispatch.

Kata Kunci: ***Polymorphism***, ***overloading***, ***overriding***, ***dynamic dispatch***, ***interface***

Daftar Pustaka

[1] Cay S. Horstmann. *Core Java Volume I–Fundamentals*. Prentice Hall, 11th edition, 2019.