POLIMORPHISM

:: Materi – 6 ::

Definisi

- ✓ Polimorfisme adalah kemampuan suatu objek untuk mengambil berbagai bentuk.
- ✓ Penggunaan polimorfisme yang paling umum di OOP terjadi ketika referensi kelas induk digunakan untuk merujuk ke objek kelas anak.
- ✓ Objek Java apa pun yang dapat lulus lebih dari satu pengujian IS-A dianggap polimorfik. Di Java, semua objek Java bersifat polimorfik karena objek apa pun untuk tipenya sendiri dan untuk kelas Object akan lulus tes IS-A.

Penting

- ✓ Untuk diketahui bahwa satu-satunya cara yang mungkin digunakan untuk mengakses suatu objek adalah melalui **variabel referensi**. Variabel referensi hanya dapat terdiri dari satu jenis. Setelah dideklarasikan, tipe variabel referensi tidak dapat diubah.
- ✓ Variabel referensi dapat dipindahkan ke objek lain asalkan tidak dinyatakan final. Tipe variabel referensi akan menentukan metode yang dapat dipanggil pada objek.
- ✓ Variabel referensi dapat merujuk ke objek apa pun dari tipe yang dideklarasikannya atau subtipe apa pun dari tipe yang dideklarasikannya. Variabel referensi juga dapat dideklarasikan sebagai kelas atau tipe *interface*.

Example


```
public interface Vegetarian {
}

public class Binatang {
}

public class Kelinci extends Binatang implements Vegetarian {
}
```

Keterangan

Sekarang, kelas Kelinci dianggap polimorfik karena memiliki banyak warisan.

Contoh berikut ini berlaku untuk Contoh di atas:

Seekor Kelinci ADALAH Binatang

Seekor Kelinci ADALAH Vegetarian

Seekor Kelinci ADALAH Seekor Kelinci

Seekor Kelinci ADALAH Objek

Penerapan

Saat menerapkan fakta bahwa variabel referensi ke referensi ke objek Kelinci maka, deklarasi objek yang dilakukan yaitu berikut ini:

Semua variabel referensi k, an, ve dan obj mengacu pada objek Kelinci yang sama di heap.

```
Kelinci k = new Kelinci();
    Animal an = k;
    vegetarian ve = k;
    object obj = k;
```

Metode Virtual

- ✓ Di bagian ini, akan ditunjukkan bagaimana perilaku metode yang diganti di Java memungkinkan untuk memanfaatkan polimorfisme saat mendesain kelas.
- ✓ Kita telah membahas penggantian metode, di mana kelas anak dapat mengganti metode pada induknya.
- ✓ Metode yang diganti pada dasarnya disembunyikan di kelas induk dan tidak dipanggil kecuali kelas anak menggunakan kata kunci ***super*** dalam metode yang diganti.

Super Keyword

Kata kunci **super** di Java adalah variabel referensi yang digunakan untuk merujuk objek kelas induk secara langsung. Setiap kali membuat subkelas turunan maka, kelas turunan dibuat secara implisit yang dirujuk oleh referensi variabel **super**.

- ✓ Super dapat digunakan untuk merujuk variabel instan kelas induk secara langsung.
- ✓ Super dapat digunakan untuk memanggil metode kelas induk secara langsung.
- ✓ super() dapat digunakan untuk memanggil konstruktor kelas induk secara langsung.

Example (class Mahasiswa)


```
public class Mahasiswa {

    private final int nim;
    private final String nama;
    private String prodi;
    private final String mk;

    //Construstor with parameters
    public Mahasiswa(int nim, String nama, String prodi, String mk) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.mk = mk;
    }

    public void mailCheck() {
        System.out.println("Mailing a check to " + this.nim + " " + this.prodi);
    }
```

Example (class Nilai)


```
public final class Nilai extends Mahasiswa {

    private int nilai;

    public Nilai(int nim, String nama, String prodi, String mk, int nilai) {
        super(nim, nama, prodi, mk);
        setNilai(nilai);
    }

    @Override
    public void mailCheck() {
        System.out.println("MailCheck Kelas Nilai");
        System.out.print("Mailing check to " + getNim() + " Nilai MK PBO " + nilai);
    }
```

Example (class NilaiDemo)


```
public class NilaiDemo {

    public static void main(String[] args) {
        Nilai n = new Nilai(20202020, "Aninda", "Teknik Informatika", "ADPL", 62);
        Mahasiswa mhs = new Nilai(20202022, "Putri", "Teknik Informatika", "ADPL", 90);
        System.out.println("\nMemanggil melalui reference Nilai --");
        n.mailCheck();
        System.out.println("\nMemanggil melalui reference Mahasiswa --");
        mhs.mailCheck();
    }
}
```

Keterangan

Pada program di atas, pada waktu kompilasi, kompiler menggunakan mailCheck() di **class Mahasiswa** untuk memvalidasi pernyataan ini. Namun, pada saat run time JVM memanggil mailCheck() di kelas **class Nilai**.

Perilaku ini disebut sebagai pemanggilan metode virtual dan metode ini disebut sebagai **metode virtual**. Metode yang diganti dipanggil pada waktu proses, apa pun tipe data referensi yang digunakan dalam kode sumber pada waktu kompilasi.

Keuntungan Polimorfisme

- ✓ Meningkatkan penggunaan kembali pada kode yang di tulis dengan mengizinkan objek dari kelas yang berbeda diperlakukan sebagai objek dari kelas yang sama.
- ✓ Meningkatkan keterbacaan dan pemeliharaan pada kode yang di tulis dengan mengurangi jumlah kode yang perlu ditulis.

Keuntungan Polimorfisme

- ✓ Mendukung pengikatan dinamis, memungkinkan metode yang benar untuk dipanggil saat *runtime* berdasarkan kelas objek yang sebenarnya.
- ✓ Memungkinkan objek diperlakukan sebagai tipe tunggal, membuatnya lebih mudah untuk menulis kode generik yang dapat menangani objek dengan tipe berbeda.

Kekurangan Polimorfisme

- ✓ Dapat mempersulit pemahaman perilaku suatu objek, terutama jika kodenya rumit atau kompleks.
- ✓ Hal ini dapat menyebabkan masalah kinerja, karena perilaku polimorfik mungkin memerlukan komputasi tambahan saat runtime

That’s all. Thank you very much! ☺

Any Questions?