PRAKTIKUM PARADIGMA PEMROGRAMAN

MODUL 11

Disusun oleh :

Nama : Fidelia Ping

NIM : 245410012

Kelas : Informatika 1

PROGRAM STUDI INFORMATIKA

PROGRAM SARJANA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

YOGYAKARTA

2025

MODUL 11
POLIMORFISME

A. TUJUAN PRAKTIKUM

Dapat membuat aplikasi yang mengimplementasi compile-time (early binding) polimorphisme, dapat membuat aplikasi yang mengimplementasi runtime (late binding) polimorphisme

B. PEMBAHASAN LISTING

PRATIKUM

-

LATIHAN

-

C. PEMBAHASAN TUGAS

Sebuah perusahaan memiliki beberapa jenis karyawan:

1. **Karyawan Tetap → ada nama dan gaji pokok + tunjungan (didapat dari 20% gajipokok)**
2. **Karyawan Kontrak → nama dan jumlah jam kerja dan upah per jam**
3. **Karyawan Magang → nama dan uang saku saja**

Setiap jenis karyawan memiliki **cara menghitung gaji yang berbeda**, tetapi semuanya memiliki **method yang sama**, yaitu hitungGaji().

Polimorfisme terjadi **satu method**, **banyak perilaku**.

Konsep Polimorfismenya :

- **Superclass**: Karyawan
- **Subclass**: KaryawanTetap, KaryawanKontrak, KaryawanMagang
- **Method** overriding
- Pemanggilan method melalui referensi superclass

STRUKTUR KELAS

Karyawan
├─ KaryawanTetap
├─ KaryawanKontrak
└─ KaryawanMagang

```
import java.util.Scanner;
class Karyawan {
  String nama;
  public Karyawan(String nama) {
    this.nama = nama;
  }
  public double hitungGaji() {
    return 0;
  }
}
class KaryawanTetap extends Karyawan {
  double gajiPokok;
  public KaryawanTetap(String nama, double gajiPokok) {
    super(nama);
    this.gajiPokok = gajiPokok;
  }
```

```
  @Override
  public double hitungGaji() {
    double tunjangan = 0.2 * gajiPokok;
    return gajiPokok + tunjangan;
  }
}
class KaryawanKontrak extends Karyawan {
   int jamKerja;
   double upahPerJam;
   public KaryawanKontrak(String nama, int jamKerja, double upahPerJam) {
    super(nama);
    this.jamKerja = jamKerja;
    this.upahPerJam = upahPerJam;
   }
   @Override
   public double hitungGaji() {
    return jamKerja * upahPerJam;
   }
}
class KaryawanMagang extends Karyawan {
   double uangSaku;
    public KaryawanMagang(String nama, double uangSaku) {
     super(nama);
     this.uangSaku = uangSaku;
    }
   @Override
   public double hitungGaji() {
    return uangSaku;
   }
}
public class SistemGaji {
   public static void main(String[] args) {
     Scanner input = new Scanner(System.in);
     Karyawan karyawan = null;
       System.out.println("=== PILIH JENIS KARYAWAN ===");
       System.out.println("1. Karyawan Tetap");
       System.out.println("2. Karyawan Kontrak");
       System.out.println("3. Karyawan Magang");
       System.out.print("Pilihan: ");
       int pilihan = input.nextInt();
       input.nextLine();
       if (pilihan == 1) {
          System.out.print("Masukkan Nama Karyawan: ");
          String nama = input.nextLine();
          System.out.print("Masukkan Gaji Pokok: ");
          double gapok = input.nextDouble();
          karyawan = new KaryawanTetap(nama, gapok);
       } else if (pilihan == 2) {
          System.out.print("Masukkan Nama Karyawan: ");
          String nama = input.nextLine();
          System.out.print("Masukkan Jam Kerja: ");
          int jam = input.nextInt();
          System.out.print("Masukkan Upah per Jam: ");
          double upah = input.nextDouble();
          karyawan = new KaryawanKontrak(nama, jam, upah);
       } else if (pilihan == 3) {
          System.out.print("Masukkan Nama Karyawan: ");
          String nama = input.nextLine();
          System.out.print("Masukkan Uang Saku: ");
```

```
            double saku = input.nextDouble();
            karyawan = new KaryawanMagang(nama, saku);
        } else {
            System.out.println("Pilihan tidak valid.");
            System.exit(0);
        }
        if (karyawan != null) {
            System.out.println("=== DATA GAJI KARYAWAN ===");
            System.out.println("Nama Karyawan : " + karyawan.nama);
            System.out.println("Total Gaji    : " + karyawan.hitungGaji());
            System.out.println("Press any key to continue . . .");
            try { System.in.read(); } catch(Exception e){}
        }
    }
}
```

Pembahasan program : Program *SistemGaji* dirancang untuk menghitung gaji karyawan dengan menerapkan konsep **polimorfisme** dalam pemrograman berorientasi objek. Class `Karyawan` berfungsi sebagai superclass yang menyimpan atribut `nama` dan method `hitungGaji()` sebagai method umum. Selanjutnya, class `KaryawanTetap`, `KaryawanKontrak`, dan `KaryawanMagang` merupakan subclass yang mewarisi class `Karyawan` dan melakukan *method overriding* pada `hitungGaji()` sesuai dengan aturan perhitungan gaji masing-masing jenis karyawan. Pada bagian `main`, program menampilkan menu pilihan jenis karyawan, menerima input dari pengguna, lalu membuat objek sesuai pilihan tersebut. Meskipun objek yang dibuat berasal dari subclass yang berbeda, semuanya disimpan dalam satu variabel bertipe `Karyawan`, sehingga program menjadi lebih fleksibel dan mudah dikembangkan tanpa perlu banyak percabangan logika.

Output

```
=== PILIH JENIS KARYAWAN ===
1. Karyawan Tetap
2. Karyawan Kontrak
3. Karyawan Magang
Pilihan: 1
Masukkan Nama Karyawan: Bunga
Masukkan Gaji Pokok: 2500000
=== DATA GAJI KARYAWAN ===
Nama Karyawan : Bunga
Total Gaji    : 3000000.0
Press any key to continue . . .
```

```
=== PILIH JENIS KARYAWAN ===
1. Karyawan Tetap
2. Karyawan Kontrak
3. Karyawan Magang
Pilihan: 2
Masukkan Nama Karyawan: Java
Masukkan Jam Kerja: 34
Masukkan Upah per Jam: 12000
=== DATA GAJI KARYAWAN ===
Nama Karyawan : Java
Total Gaji    : 408000.0
Press any key to continue . . .
```

```
=== PILIH JENIS KARYAWAN ===
1. Karyawan Tetap
2. Karyawan Kontrak
3. Karyawan Magang
Pilihan: 3
Masukkan Nama Karyawan: Avril
Masukkan Uang Saku: 450000
=== DATA GAJI KARYAWAN ===
Nama Karyawan : Avril
Total Gaji    : 450000.0
Press any key to continue . . .
```

Pembahasan output : Output program diawali dengan tampilan menu pemilihan jenis karyawan, kemudian pengguna diminta memasukkan data sesuai dengan pilihan yang dipilih, seperti nama dan gaji pokok untuk karyawan tetap, jam kerja dan upah per jam untuk karyawan kontrak, atau uang saku untuk karyawan magang. Setelah data dimasukkan, program menampilkan hasil berupa nama karyawan dan total gaji yang telah dihitung. Nilai total gaji yang muncul berbeda-beda tergantung jenis karyawan, karena method `hitungGaji()` yang dijalankan menyesuaikan dengan objek yang dibuat. Hal ini membuktikan bahwa proses perhitungan gaji berjalan dengan benar dan menunjukkan penerapan polimorfisme, di mana satu method yang sama menghasilkan output yang berbeda sesuai tipe objek karyawan.

D. KESIMPULAN

Berdasarkan hasil praktikum polimorfisme, dapat disimpulkan bahwa konsep polimorfisme dalam pemrograman berorientasi objek memungkinkan satu metode yang sama digunakan oleh objek yang berbeda dengan perilaku yang berbeda pula. Melalui penerapan **method overriding** dan penggunaan **reference class induk (parent)** yang menunjuk ke objek class turunan (child), program menjadi lebih fleksibel, mudah dikembangkan, dan terstruktur. Praktikum ini menunjukkan bahwa polimorfisme membantu mengurangi ketergantungan terhadap tipe objek secara spesifik, meningkatkan keterbacaan kode, serta mendukung prinsip *extensibility* dalam pengembangan perangkat lunak. Dengan demikian, polimorfisme berperan penting dalam penerapan desain program yang efisien, modular, dan sesuai dengan prinsip pemrograman berorientasi objek.