PRAKTIKUM PARADIGMA PEMROGRAMAN

MODUL 8

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 8
PEWARISAN

A. TUJUAN PRAKTIKUM

1. Dapat mendefinisikan kelas turunan, Dapat membuat Objek turunan, Dapat mengakses atribut turunan
2. Dapat menerapkan kons-truktor pada sub kelas dalam pewarisan.

B. PEMBAHASAN LISTING

PRATIKUM

-

LATIHAN

-

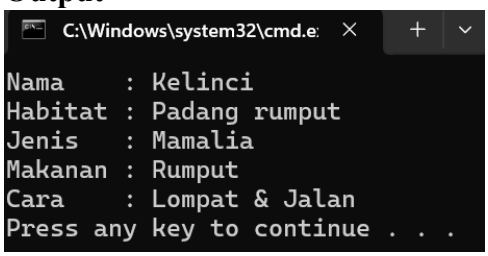
C. PEMBAHASAN TUGAS

```
class Hewan {
    protected String nama;
    protected String habitat;

    public Hewan(String nama, String habitat) {
        this.nama = nama;
        this.habitat = habitat;
    }

    public void tampilInfo() {
        System.out.println("Nama    : " + nama);
        System.out.println("Habitat : " + habitat);
    }
}

class Jenis extends Hewan {
    protected String jenis;

    public Jenis(String nama, String habitat, String jenis) {
        super(nama, habitat);
        this.jenis = jenis;
    }

    @Override
    public void tampilInfo() {
        super.tampilInfo();
        System.out.println("Jenis   : " + jenis);
    }
}

class Ciri extends Jenis {
    private String makanan;
    private String caraBergerak;

    public Ciri(String nama, String habitat, String jenis,
                String makanan, String caraBergerak) {
        super(nama, habitat, jenis);
        this.makanan = makanan;
        this.caraBergerak = caraBergerak;
    }
```

```
    @Override
    public void tampilInfo() {
        super.tampilInfo();
        System.out.println("Makanan : " + makanan);
        System.out.println("Cara    : " + caraBergerak);
    }
}

public class Kelinci {
    public static void main(String[] args) {
        Ciri k = new Ciri("Kelinci", "Padang rumput", "Mamalia",
                          "Rumput", "Lompat & Jalan");
        k.tampilInfo();
    }
}
```

Pembahasan program : Program tersebut mendefinisikan sebuah hierarki kelas di mana Hewan berfungsi sebagai superclass dasar yang menyimpan informasi umum seperti `nama` dan `habitat`; kemudian subclass Jenis memperluas Hewan dengan menambahkan atribut `jenis`, dan subclass berikutnya Ciri kembali memperluas Jenis dengan atribut tambahan misalnya `makanan` dan `caraBergerak` . Dalam subclass-subclass ini, konstruktor menggunakan keyword `super(...)` untuk memanggil konstruktor superclass agar bagian atribut dari Hewan (nama, habitat) diinisialisasi dengan benar — ini memungkinkan pewarisan sifat dasar tanpa perlu menulis ulang kode. Kemudian, method `tampilInfo()` di setiap kelas menggunakan `super.tampilInfo()` untuk menjalankan versi superclass sebelum menambahkan informasi kelas sendiri — memanfaatkan method overriding untuk memperluas perilaku tanpa menggantinya total. Dengan demikian, program ini secara benar menerapkan prinsip inheritance, reuse kode, dan memperlihatkan bagaimana keyword `super` membantu menjaga struktur hierarki kelas serta konsistensi data dalam objek.

Output



Pembahasan output :

- Nama : Kelinci — ini berasal dari method tampilInfo() di Hewan, menampilkan nilai nama (yang di-set lewat constructor Ciri, diteruskan ke Hewan via super).
- Habitat : Padang rumput — juga dari Hewan, menampilkan nilai habitat.
- Jenis : Mamalia — dari override tampilInfo() di Jenis: setelah super.tampilInfo(), ditambahkan info jenis.
- Makanan : Rumput — dari override tampilInfo() di Ciri: menampilkan nilai atribut makanan yang unik untuk kelas Ciri.
- Cara : Lompat & Jalan — dari Ciri, menampilkan nilai caraBergerak, menunjukkan karakteristik tambahan objek Ciri.

D. KESIMPULAN

Saya menyimpulkan bahwa dengan menggunakan konsep pewarisan di Java yaitu memanfaatkan superclass (kelas dasar) dan subclass (kelas turunan) serta keyword super saya dapat membuat struktur kode yang lebih rapi, efisien, dan mudah untuk dikembangkan. Karena superclass (misalnya Hewan) menyediakan atribut dan metode dasar (seperti nama dan habitat), saya bisa membuat kelas-kelas turunan (misalnya Jenis dan Ciri) yang mewarisi bagian tersebut, kemudian menambahkan detail spesifik (jenis, makanan, cara bergerak) tanpa harus menulis ulang kode dasar ini ilustrasi dari prinsip *code reusability*. Kemudian melalui super(...) di konstruktor dan super.tampilInfo() di method override, saya memastikan bagian umum tercakup dahulu sebelum menambahkan detail di subclass membuat program lebih modular dan mudah dipelihara. Secara keseluruhan, pewarisan membantu saya membangun hierarki kelas yang logis dan fleksibel, memungkinkan reuse, ekstensi, dan menghindari duplikasi kode ketika membuat banyak jenis objek yang berbagi sifat dasar.