PRAKTIKUM PARADIGMA PEMROGRAMAN

MODUL 10

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 10
INTERFACE

A. TUJUAN PRAKTIKUM

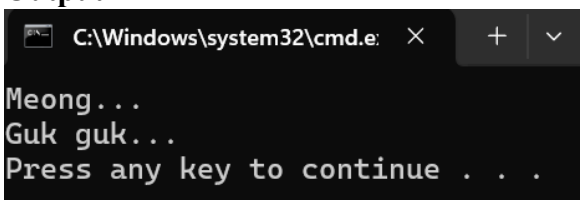
1. Mengatur Standar / Kontrak → Interface memaksa class memiliki pola yang sama. Misal semua hewan harus punya method suara().
2. Mendukung Multiple Inheritance → Java hanya bisa mewarisi 1 class (single inheritance), tapi bisa implements banyak interface.
3. Memisahkan "apa yang dilakukan" dari "bagaimana cara melakukannya"Interface → apa yang harus ada Class → bagaimana cara kerjanya
4. Mempermudah pembuatan program yang konsisten dan mudah dikembangkan Cocok untuk aplikasi besar, tim kerja, dan framework.
5. Untuk Polymorphism → Objek bisa dipanggil lewat referensi interface.

B. PEMBAHASAN LISTING PRATIKUM

```
interface Hewan {
    void suara(); // method tanpa isi
}
class Kucing implements Hewan {
    public void suara() {
        System.out.println("Meong...");
    }
}

class Anjing implements Hewan {
    public void suara() {
        System.out.println("Guk guk...");
    }
}
public class Main {
    public static void main(String[] args) {
        Hewan h1 = new Kucing();
        Hewan h2 = new Anjing();

        h1.suara();
        h2.suara();
    }
}
```

Pembahasan program : Program ini menggunakan konsep *interface* dan *polymorphism* dalam Java. Interface **Hewan** mendefinisikan method `suara()` yang harus diimplementasikan oleh setiap class yang menggunakannya. Class **Kucing** dan **Anjing** mengimplementasikan interface tersebut dengan memberikan versi suara masing-masing, yaitu "Meong..." untuk kucing dan "Guk guk..." untuk anjing. Pada method `main`, variabel bertipe **Hewan** digunakan untuk menyimpan objek **Kucing** dan **Anjing**, sehingga ketika `suara()` dipanggil, Java otomatis menjalankan method sesuai objek yang sebenarnya. Hal ini menunjukkan penerapan polymorphism, di mana satu tipe referensi (`Hewan`) dapat merujuk ke berbagai objek berbeda.

Output



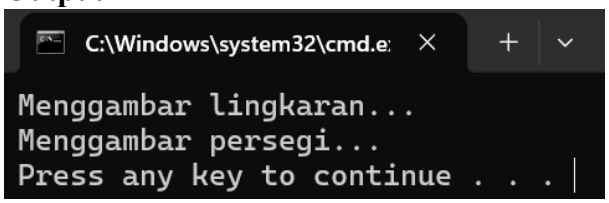
Pembahasan output : Output menampilkan suara dari masing-masing hewan berdasarkan objek yang digunakan. Ketika pemanggilan `h1.suara()` dilakukan, karena `h1` berisi objek **Kucing**, maka program mencetak "Meong...". Selanjutnya pemanggilan `h2.suara()` menampilkan "Guk guk..." karena `h2` adalah objek **Anjing**. Dengan demikian output mengikuti implementasi method pada class masing-masing hewan.

LATIHAN

```
interface Bentuk {
    void gambar();
}

class Lingkaran implements Bentuk {
    public void gambar() {
        System.out.println("Menggambar lingkaran...");
    }
}

class Persegi implements Bentuk {
    public void gambar() {
        System.out.println("Menggambar persegi...");
    }
}

public class DemoPolymorphism {
    public static void main(String[] args) {
        Bentuk b;

        b = new Lingkaran();
        b.gambar();

        b = new Persegi();
        b.gambar();
    }
}
```

Pembahasan program : Program ini menunjukkan penggunaan interface dan konsep polymorphism dalam Java. Interface **Bentuk** mendeklarasikan method `gambar()` yang harus diimplementasikan oleh setiap class yang menggunakannya. Class **Lingkaran** dan **Persegi** mengimplementasikan interface tersebut dengan menyediakan versi method `gambar()` masing-masing yang menampilkan jenis bentuk yang digambar. Pada method `main`, variabel bertipe **Bentuk** digunakan untuk menyimpan objek dari dua class berbeda, yaitu **Lingkaran** dan **Persegi**. Saat method `gambar()` dipanggil, Java akan mengeksekusi method sesuai objek yang sedang disimpan pada variabel `b`. Hal ini menunjukkan polymorphism, yaitu kemampuan satu tipe referensi untuk merujuk ke berbagai objek berbeda.

Output



Pembahasan output : Output program menampilkan dua baris yang menunjukkan hasil pemanggilan method `gambar()` dari dua objek yang berbeda. Ketika variabel `b` diisi objek **Lingkaran** dan dipanggil `b.gambar()`, program mencetak **"Menggambar lingkaran..."**. Setelah itu, variabel `b` diisi ulang dengan objek **Persegi**, sehingga pemanggilan `b.gambar()` menghasilkan **"Menggambar persegi..."**. Output mengikuti implementasi method pada masing-masing class.

C. PEMBAHASAN TUGAS

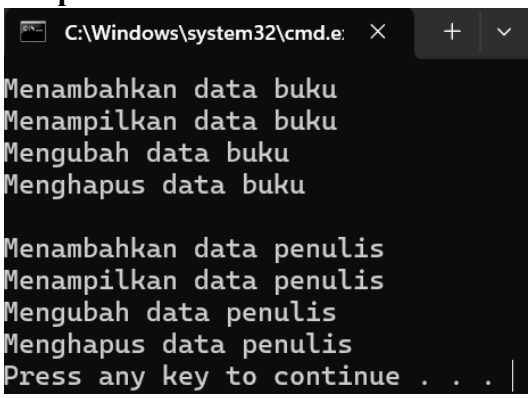
CRUD BUKU & PENULIS

```
interface Crud {
    void tambah();
    void tampil();
    void ubah();
    void hapus();
}
class CrudBuku implements Crud {
    public void tambah() {
        System.out.println("Menambahkan data buku");
    }
    public void tampil() {
        System.out.println("Menampilkan data buku");
    }
    public void ubah() {
        System.out.println("Mengubah data buku");
    }
    public void hapus() {
        System.out.println("Menghapus data buku");
    }
}
class CrudPenulis implements Crud {
    public void tambah() {
        System.out.println("Menambahkan data penulis");
    }
    public void tampil() {
        System.out.println("Menampilkan data penulis");
    }
    public void ubah() {
        System.out.println("Mengubah data penulis");
    }
    public void hapus() {
        System.out.println("Menghapus data penulis");
    }
}
public class DemoCRUD {
    public static void main(String[] args) {
        Crud c;
        c = new CrudBuku();
        c.tambah();
        c.tampil();
```

```
        c.ubah();
        c.hapus();
   System.out.println();
        c = new CrudPenulis();
        c.tambah();
        c.tampil();
        c.ubah();
        c.hapus();
    }
}
```

Pembahasan program : Program ini menggunakan interface **Crud** untuk mendefinisikan empat operasi dasar: `tambah()`, `tampil()`, `ubah()`, dan `hapus()`. Interface ini kemudian diimplementasikan oleh dua class, yaitu **CrudBuku** dan **CrudPenulis**, di mana masing-masing class memberikan isi method sesuai konteksnya. Pada class `DemoCRUD`, digunakan variabel bertipe `Crud` untuk menampung objek dari dua implementasi berbeda. Hal ini menunjukkan penerapan polymorphism, karena variabel `c` dapat berisi objek `CrudBuku` maupun `CrudPenulis`. Ketika method CRUD dipanggil melalui variabel `c`, program menjalankan versi method sesuai objek yang sedang digunakan, meskipun tipe referensinya sama, yaitu `Crud`.

Output



Pembahasan output : Output program pertama-tama menampilkan serangkaian operasi CRUD untuk objek **CrudBuku**, yaitu: "Menambahkan data buku", "Menampilkan data buku", "Mengubah data buku", dan "Menghapus data buku". Setelah itu ditampilkan baris kosong sebagai pemisah. Kemudian program berpindah ke objek **CrudPenulis** dan mencetak operasi serupa tetapi dengan konteks "penulis": "Menambahkan data penulis", "Menampilkan data penulis", "Mengubah data penulis", dan "Menghapus data penulis". Output ini menunjukkan bahwa method yang dipanggil mengikuti implementasi yang ada pada masing-masing class.

D. KESIMPULAN

Dari praktikum ini dapat disimpulkan bahwa **interface** merupakan fitur penting dalam pemrograman Java yang digunakan untuk mendefinisikan kumpulan method tanpa isi (abstrak) sehingga class lain dapat mengimplementasikannya sesuai kebutuhan. Dengan menggunakan interface, program menjadi lebih terstruktur, fleksibel, dan mudah dikembangkan. Interface juga memungkinkan penerapan **polymorphism**, yaitu kemampuan satu tipe referensi untuk memanggil method dari berbagai objek yang berbeda implementasinya. Melalui praktikum ini, mahasiswa dapat memahami bagaimana interface membantu menciptakan standar perilaku pada berbagai class,

meningkatkan modularitas kode, serta mempermudah proses pengelolaan dan pengembangan aplikasi.