# RESPONSI PEMROGRAMAN BERORIENTASI OBJEK PRAKTIK


PROGRAM STUDI INFORMATIKA FAKULTAS SAINS & TEKNOLOGI UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024

SOAL TEORI RESPONSI :

1. Jelaskan perbedaan use case diagram dengan class diagram?
2. Jelaskan jenis-jenis dependensi?
3. Apa perbedaan pemrograman terstruktur dengan berorientasi objek,jelaskan?
4. Jelaskan konsep objek dan beri contohnya?
5. Jelaskan jenis-jenis acces modifier, beri contohnya dalam baris pemrograman?
6. Gambarkan contoh pewarisan dalam diagram class?

1. menjelaskan perbedaan use case diagram dengan class diagram? Use Case Diagram : Merupakan diagram yang menggambarkan interaksi antara aktor (user) dengan sistem. Diagram ini menunjukkan bagaimana pengguna menggunakan sistem untuk mencapai tujuannya. Class Diagram : Merupakan diagram yang menggambarkan struktur dan hubungan antar kelas dalam sistem. Diagram ini menunjukkan kelas, atribut, dan metode yang dimiliki oleh setiap kelas, serta hubungan antar kelas seperti pewarisan dan asosiasi.

2. menjelaskan jenis-jenis dependensi?

Dalam pemodelan perangkat lunak, terutama dalam konteks diagram kelas dan desain berorientasi objek, dependensi merujuk pada hubungan antara dua elemen di mana satu elemen (klien) bergantung pada elemen lain (supplier) untuk berfungsi dengan baik. Berikut adalah beberapa jenis dependensi yang umum dalam pemodelan perangkat lunak:

**Dependensi Keterhubungan (Association)**

- Deskripsi: Hubungan antara dua kelas di mana satu kelas menggunakan atau berinteraksi dengan kelas lainnya.
- Contoh: Sebuah kelas Mahasiswa dapat memiliki hubungan dengan kelas MataKuliah, di mana seorang mahasiswa terdaftar dalam beberapa mata kuliah.

**Dependensi Agregasi (Aggregation)**

- Deskripsi: Hubungan "bagian-dari" yang menunjukkan bahwa satu objek terdiri dari beberapa objek lain, tetapi objek-objek tersebut dapat eksis secara independen.

- Contoh: Kelas Departemen dapat memiliki agregasi dengan kelas Dosen, di mana seorang dosen bisa saja tidak terikat pada departemen tertentu.

**Dependensi Komposisi (Composition)**

- Deskripsi: Hubungan "bagian-dari" yang lebih kuat daripada agregasi, di mana bagian tidak dapat eksis tanpa keseluruhan. Jika objek keseluruhan dihancurkan, maka bagian-bagiannya juga akan dihancurkan.
- Contoh: Kelas Mobil dapat memiliki komposisi dengan kelas Mesin, di mana mesin tidak dapat ada tanpa mobil.

**Dependensi Realisasi (Realization)**

- Deskripsi: Hubungan antara antarmuka (interface) dan kelas yang mengimplementasikannya. Kelas yang mengimplementasikan antarmuka harus menyediakan implementasi untuk semua metode yang didefinisikan dalam antarmuka.
- Contoh: Antarmuka Pembayaran dapat direalisasikan oleh kelas KartuKredit dan TransferBank.

**Dependensi Penggunaan (Usage)**

- Deskripsi: Hubungan di mana satu kelas menggunakan kelas lain sebagai parameter, atau dalam metode yang diimplementasikan. Ini menunjukkan bahwa perubahan pada kelas yang digunakan dapat mempengaruhi kelas yang menggunakannya.
- Contoh: Kelas Order yang memiliki metode proses(Pembayaran pembayaran) menunjukkan bahwa Order bergantung pada kelas Pembayaran.

**Dependensi Ketergantungan (Dependency)**

- Deskripsi: Hubungan di mana satu kelas bergantung pada kelas lain untuk berfungsi. Jika kelas yang digunakan berubah, kelas yang bergantung mungkin juga perlu diperbarui.
- Contoh: Kelas Pengguna yang memiliki metode login(Kredensial kredensial), di mana Pengguna bergantung pada kelas Kredensial.

**Dependensi Asosiasi Berarah (Directed Association)**

- Deskripsi: Hubungan di mana satu kelas memiliki referensi ke kelas lain, tetapi tidak sebaliknya. Ini menunjukkan arah ketergantungan.
- Contoh: Kelas Dokter dapat memiliki asosiasi berarah ke kelas Pasien, di mana dokter merawat pasien, tetapi pasien tidak perlu mengetahui dokter mana yang merawatnya.

3. Apa perbedaan pemrograman yang terstruktur dengan orientasi objek, jelaskan?

Pemrograman Terstruktur :

- Fokus pada urutan proses (alur) dalam menyelesaikan masalah.
- Digunakan untuk memecah masalah menjadi fungsi-fungsi kecil.
- Mengutamakan proses daripada data.
- Contoh: Bahasa C, Pascal

Pemrograman Berorientasi Objek :

- Fokus pada objek-objek dan interaksi antar objek.
- Menggabungkan data dan fungsi dalam satu kesatuan (objek).
- Mengutamakan data dan manipulasi data.
- Contoh: Java, Python, C++

4.Jelaskan konsep objek dan beri contohnya?
Objek adalah instance dari kelas yang merepresentasikan entitas nyata dengan atribut (data) dan perilaku (metode). Misalnya, dalam program untuk perpustakaan,

kelas Buku dapat memiliki objek buku1 yang mewakili buku tertentu dengan atribut seperti judul, pengarang, dan tahunTerbit, serta metode seperti pinjam() dan kembalikan().

5.Access modifier berfungsi untuk mengatur hak akses terhadap suatu anggota kelas (method, field, dan lain-lain). Terdapat beberapa jenis pengubah akses:

- Publik: Anggota kelas yang dideklarasikan dengan publik dapat diakses dari mana saja, baik dari dalam kelas itu sendiri, kelas turunan, maupun dari kelas lain yang berada dalam paket yang sama atau paket berbeda.
- Protected: Anggota kelas yang dideklarasikan dengan protected dapat diakses dari dalam kelas itu sendiri, kelas turunan, dan kelas lain yang berada dalam paket yang sama.
- Private: Anggota kelas yang dideklarasikan dengan private hanya dapat diakses dari dalam kelas itu sendiri.
- Default (package private): Anggota kelas yang tidak dideklarasikan dengan access modifier tertentu (public, protected, atau private) disebut dengan default access modifier. Anggota kelas dengan default access modifier dapat diakses dari dalam kelas itu sendiri, kelas lain yang berada dalam paket yang sama, tetapi tidak dapat diakses dari kelas lain yang berada di paket yang berbeda.

```
public class MyClass {



    // Field dengan access modifier public

    public int publicField;



    // Method dengan access modifier protected

    protected void protectedMethod() {

        // ...

    }
```

```
    // Method dengan access modifier private
    private void privateMethod() {
        // ...
    }

    // Method dengan access modifier default (package private)
    void defaultMethod() {
        // ...
    }
}
```

Pada contoh di atas:

- **`publicField`** dapat diakses dari mana saja.
- **`protectedMethod()`** dapat diakses dari **`MyClass`** itu sendiri, kelas turunan, dan kelas lain yang berada dalam paket yang sama.
- **`privateMethod()`** hanya dapat diakses dari **`MyClass`** itu sendiri.
- **`defaultMethod()`** dapat diakses dari **`MyClass`** itu sendiri dan kelas lain yang berada dalam paket yang sama.

Access modifier sangat penting dalam pemrograman yang berorientasi pada objek untuk menjaga keamanan dan integritas data. Dengan menggunakan access modifier, kita dapat mengatur bagaimana anggota kelas diakses dan digunakan.

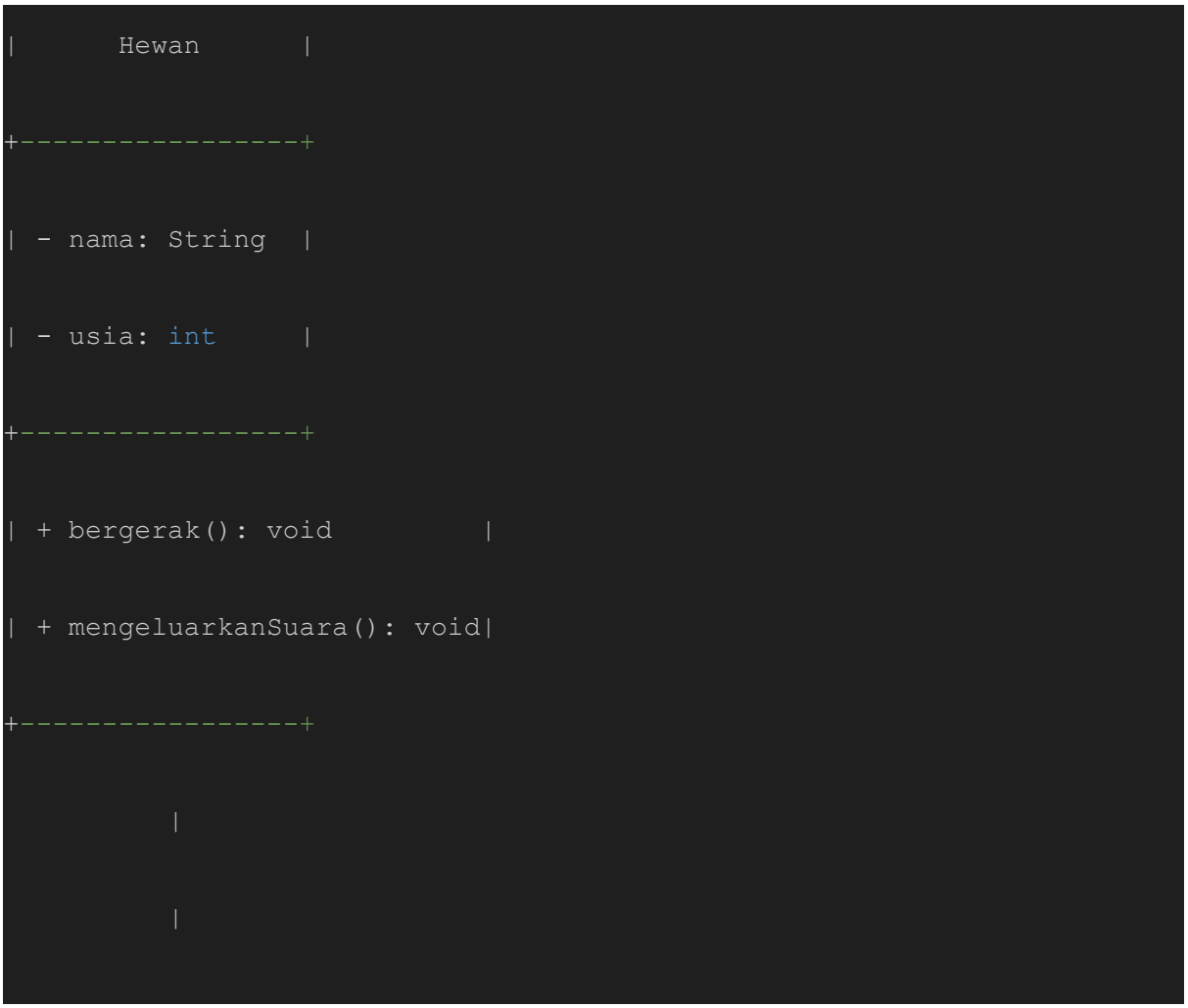
6.Berikut adalah contoh sederhana diagram class untuk menggambarkan pewarisan (inheritance) dalam pemrograman berorientasi objek. Dalam contoh ini, kita memiliki tiga kelas:

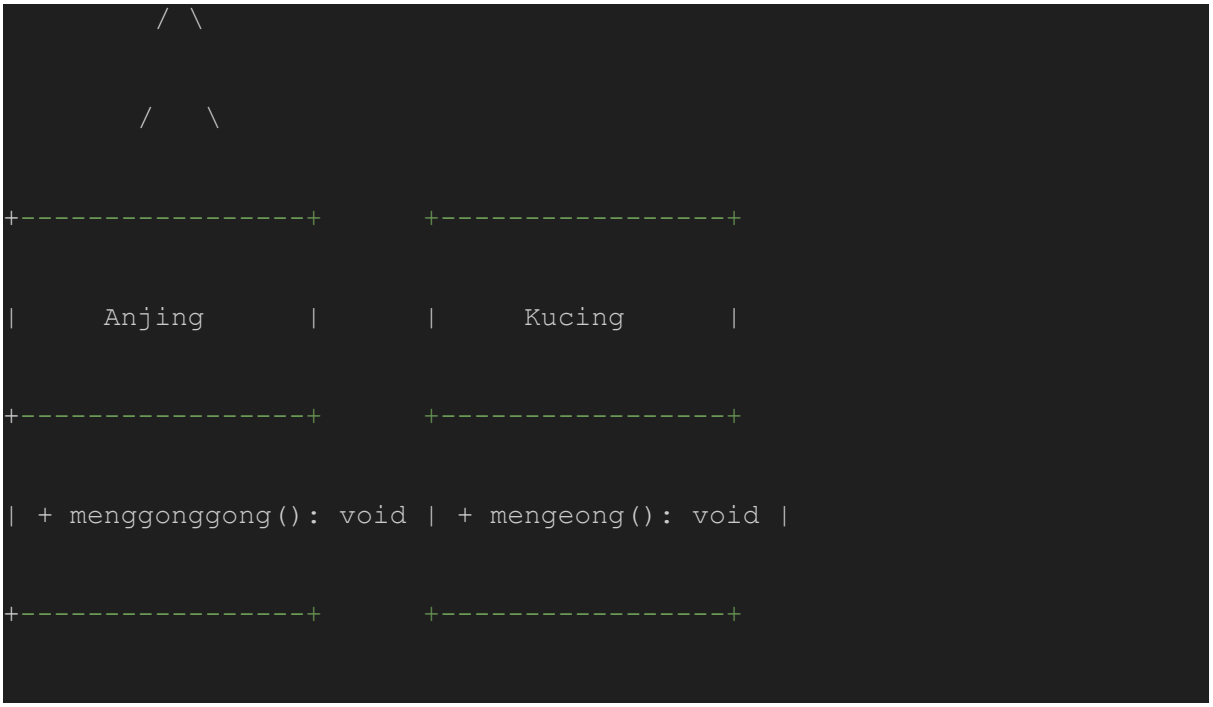
1. Kelas Induk: Hewan
2. Kelas Turunan: Anjing dan Kucing

Deskripsi Struktur Pewarisan:

- Hewan adalah kelas induk yang memiliki atribut umum seperti `nama` dan `usia`, serta metode umum seperti `bergerak()` dan `mengeluarkanSuara()`.
- `Anjing` dan `Kucing` adalah kelas turunan yang mewarisi atribut dan metode dari kelas Hewan. Mereka juga dapat memiliki metode khusus seperti `menggonggong()` untuk `Anjing` dan `mengeong()` untuk `Kucing`.

Berikut adalah diagram class yang menggambarkan pewarisan ini:






### Penjelasan Diagram:

- **Kelas Hewan** adalah kelas induk yang memiliki atribut `nama` dan `usia`, serta **metode `bergerak()`** dan **`mengeluarkanSuara()`**.
- **Kelas `Anjing`** dan **Kelas `Kucing`** mewarisi semua atribut dan metode dari kelas Hewan, tetapi masing-masing menambahkan metode spesifik: **`menggonggong()`** untuk **`Anjing`** dan **`mengeong()`** untuk `Kucing`.

Diagram ini menunjukkan bagaimana pewarisan memungkinkan kelas turunan untuk memiliki sifat-sifat kelas induk sambil tetap dapat menambahkan atau mengubah perilaku mereka sendiri.