LAPORAN PEMROGRAMAN BERBASIS OBJEK
“RESPONSI”

AMARULLAH ARSADINATA
5230411323

PROGRAM STUDI INFORMATIKA FALKUSTAS SAINS DAN TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
TAHUN AJARAN
2024/2025

SOAL TEORI

1. Jelaskan perbedaan use case dingram 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 access modifier beri contohnya dalam baris pemrograman?
6. Gambarkan contoh pewarisan dalam diagram class?

Berikut adalah jawaban dari pertanyaan yang diberikan:

1. Perbedaan Use Case Diagram dengan Class Diagram

- **Use Case Diagram**: Use case diagram menggambarkan interaksi antara aktor (pengguna atau sistem lain) dan sistem dalam memenuhi suatu fungsi atau layanan. Diagram ini digunakan untuk menggambarkan fungsi yang dimiliki oleh sistem dan bagaimana aktor berinteraksi dengan fungsi tersebut. Biasanya digunakan pada tahap awal pengembangan sistem untuk memahami kebutuhan pengguna.

  Contoh: Sistem E-commerce, use case seperti "Melakukan Pembelian", "Login", "Mencari Produk", dll.

- **Class Diagram**: Class diagram menggambarkan struktur kelas dalam sistem, termasuk atribut, metode, dan hubungan antar kelas. Diagram ini menunjukkan struktur statis dari sistem dan bagaimana objek-objek dalam sistem saling berhubungan, seperti melalui asosiasi, pewarisan (inheritance), agregasi, atau komposisi. Biasanya digunakan untuk desain detail dari arsitektur sistem.

  Contoh: Sistem E-commerce, class seperti "User", "Produk", "Keranjang", dll.

2. Jenis-jenis Dependensi

a. **Dependensi Langsung (Direct Dependency)**
- Dependensi ini adalah komponen atau pustaka yang secara langsung dibutuhkan oleh modul atau komponen tertentu. Contohnya, jika aplikasi web menggunakan pustaka axios untuk mengirim permintaan HTTP, maka axios adalah dependensi langsung aplikasi tersebut.

b. **Dependensi Tidak Langsung (Indirect Dependency)**
- Ini adalah dependensi dari dependensi lain. Misalnya, jika aplikasi menggunakan pustaka A yang pada gilirannya menggunakan pustaka B, maka pustaka B adalah dependensi tidak langsung dari aplikasi tersebut.

c. **Dependensi Waktu (Temporal Dependency)**
- Dependensi ini terjadi ketika satu proses harus diselesaikan sebelum proses lain dimulai.

Misalnya, dalam pipeline pemrosesan data, proses pembersihan data harus selesai sebelum proses analisis data dimulai.

d. **Dependensi Fungsional (Functional Dependency)**
- Ini mengacu pada situasi di mana fungsi atau modul tertentu membutuhkan data atau layanan dari fungsi lain. Dalam basis data, dependensi ini sering merujuk pada hubungan antara kolom yang menentukan nilai kolom lain dalam tabel.

e. **Dependensi Eksternal (External Dependency)**
- Dependensi ini melibatkan komponen atau layanan dari luar sistem, seperti layanan API eksternal atau pustaka pihak ketiga. Aplikasi mungkin memerlukan akses ke API eksternal untuk mengakses data atau fungsi tertentu.

f. **Dependensi Waktu Kompilasi (Compile-time Dependency)**
- Ini adalah dependensi yang diperlukan saat proses kompilasi kode. Contohnya adalah pustaka yang harus diimpor agar kode dapat berhasil dikompilasi.

g. **Dependensi Waktu Eksekusi (Runtime Dependency)**
- Dependensi ini dibutuhkan selama eksekusi aplikasi. Contohnya adalah pustaka yang hanya digunakan saat aplikasi berjalan, seperti pustaka logging yang diinisialisasi saat runtime.

h. **Dependensi Logis (Logical Dependency)**
- Ini melibatkan aturan atau logika yang menentukan urutan pelaksanaan tugas atau fungsi. Misalnya, jika ada dua tugas yang terkait secara logis, tugas pertama harus selesai sebelum tugas kedua dapat dimulai.

3. Perbedaan Pemrograman Terstruktur dengan Pemrograman Berorientasi Objek

- **Pemrograman Terstruktur**:
  - Berfokus pada alur program dan pemrosesan data melalui fungsi atau prosedur.
  - Menggunakan pendekatan top-down dalam pengembangan kode, di mana program dipecah menjadi fungsi-fungsi yang lebih kecil.
  - Tidak ada konsep objek dan kelas.
  - **Contoh Bahasa**: C, Pascal.

- **Pemrograman Berorientasi Objek (OOP)**:
  - Berfokus pada pengelompokan data dan fungsionalitasnya ke dalam objek.
  - Menggunakan konsep kelas dan objek untuk menyusun program, yang membuatnya lebih modular dan mudah dikelola.
  - Menerapkan prinsip OOP seperti enkapsulasi, pewarisan, polimorfisme, dan abstraksi.
  - **Contoh Bahasa**: Java, Python, C++.

4. Konsep Objek dan Contohnya

- **Objek**: Objek adalah instance dari sebuah kelas dan merupakan entitas yang memiliki atribut (properti/data) dan perilaku (metode). Dalam pemrograman, objek merepresentasikan sesuatu yang nyata (seperti mobil, manusia) atau abstrak (seperti transaksi, laporan). Objek digunakan untuk menyimpan data dan metode yang beroperasi pada data tersebut.

Contoh:
python

```python
class Mobil:
    def __init__(self, merk, warna):
        self.merk = merk
        self.warna = warna

    def nyalakan(self):
        print(f"{self.merk} berwarna {self.warna} sedang menyala.")

mobil_saya = Mobil("Toyota", "Merah")
mobil_saya.nyalakan()
```

5. Jenis-jenis Access Modifier dan Contohnya dalam Pemrograman

- **Public**: Dapat diakses dari mana saja, baik dari dalam maupun luar kelas.
- **Protected**: Hanya dapat diakses oleh kelas itu sendiri dan kelas turunannya (subclass).
- **Private**: Hanya dapat diakses dari dalam kelas itu sendiri, tidak bisa diakses dari luar kelas atau oleh subclass.

Contoh (dalam Python):
python

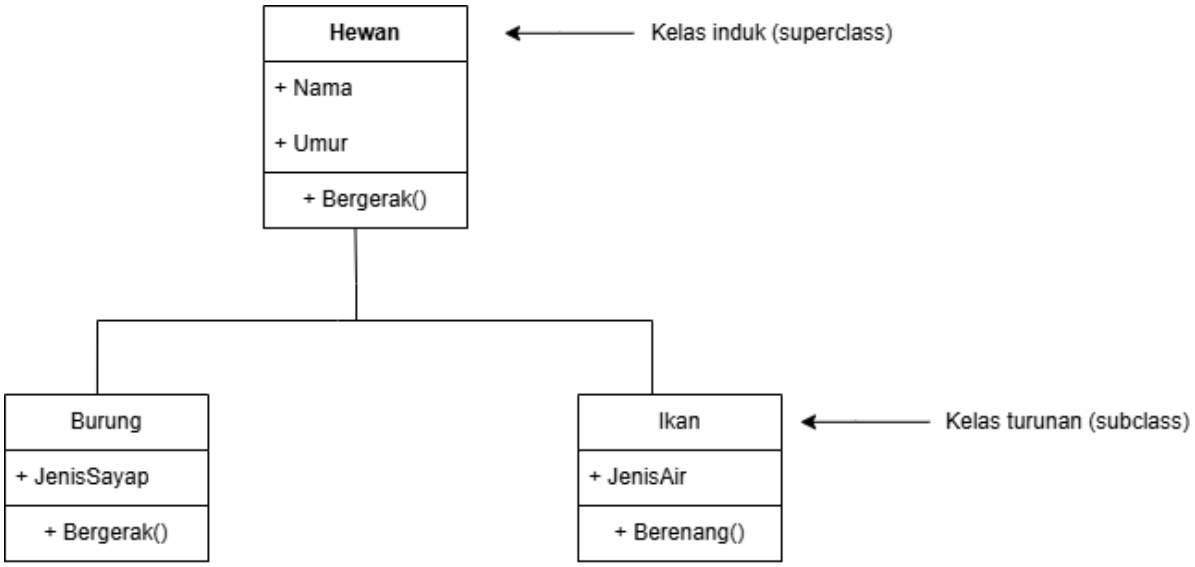
```python
class Contoh:
    def __init__(self):
        self.public = "Ini public"           # Public
        self._protected = "Ini protected"    # Protected (Conventionally with
underscore)
        self.__private = "Ini private"       # Private (Double underscore)

    def tampilkan(self):
        print(self.public)
        print(self._protected)
        print(self.__private)

obj = Contoh()
print(obj.public)         # Bisa diakses
print(obj._protected)     # Bisa diakses, tetapi hanya sebagai konvensi
# print(obj.__private)    # Tidak bisa diakses langsung dari luar
```

6. Contoh Pewarisan dalam Class Diagram

Berikut adalah contoh pewarisan dalam diagram kelas untuk konsep hewan dengan beberapa turunan:



Pada diagram di atas:
- Kelas Hewan merupakan superclass yang memiliki atribut umum nama dan umur, serta metode bergerak().
- Kelas Burung dan Ikan mewarisi kelas Hewan. Burung memiliki atribut tambahan jenisSayap dan metode terbang(), sedangkan Ikan memiliki atribut jenisAir dan metode berenang().