Nama : Arif Nur Rahman
NPM : 5230411228

# RESNPONSI PBO

## Soal Teori

1.Jelaskan perbedaan use case diagram dengan class diagram?

Use case diagram digunakan untuk menunjukkan interaksi antara pengguna (User) dan sistem. Diagram ini menunjukkan bagaimana User berinteraksi dengan sistem saat menjalankan program dengan komputer.

Class diagram digunakan untuk menggambarkan struktur dasar dari sistem mulai dari kelas-kelas yang ada dalam sistem, atribut, metode, dan hubungan antar kelas.

2.Jelaskan jenis-jenis dependensi?

Dependensi adalah hubungan antara elemen-elemen dalam sistem, di mana satu elemen bergantung pada elemen lain.Terdapat beberapa jenis dependensi :

- Class Dependency:Yaitu ketika satu kelas bergantung pada kelas lain untuk fungsi atau data.
- Composition Dependency : Hubungan di mana satu kelas (komponen) adalah bagian dari kelas lain (komposit). Jika komposit dihapus, maka komponen juga dihapus.
- Aggregation Dependency :Hubungan di mana satu kelas (agregat) berisi kelas lain (komponen), tetapi komponen dapat eksis secara independen dari agregat.
- Association Dependency :Hubungan antara dua kelas di mana satu kelas menggunakan atau berinteraksi dengan kelas lain.
- Inheritance Dependency : Hubungan di mana satu kelas (subkelas) mewarisi atribut dan metode dari kelas lain (superkelas).
- Realization Dependency :Hubungan di mana suatu kelas mengimplementasikan antarmuka (interface).
- Temporal Dependency :Hubungan di mana satu elemen harus ada atau dilakukan sebelum elemen lain.
- Performance Dependency :Hubungan di mana kinerja satu komponen dapat mempengaruhi kinerja komponen lain.

3.Apa perbedaan pemrograman terstruktur dengan berorientasi objek,jelaskan?

- Pemrograman Terstruktur:
  Berfokus pada pengorganisasian kode dalam bentuk fungsi atau prosedur. Tujuannya adalah untuk meningkatkan efisieansi dan pemeliharaan kode dengan memecah program menjadi bagian-bagian yang lebih kecil dan terstruktur.
- Pemrograman Berorientasi Objek :
  Berfokus pada objek sebagai dasar dari pemrograman. Objek adalah kombinasi dari data dan fungsi yang beroperasi pada data tersebut.Tujuannya adalah untuk menciptakan model lebih mudah dipahami dan diorganisir, serta untuk meningkatkan pemanfaatan kode dan pemeliharaan.

4.Jelaskan konsep objek dan beri contohnya?

Konsep objek adalah salah satu prinsip dasar dalam PBO.Objek adalah entitas yang memiliki atribut dan metode yang beroperasi pada data tersebut. Dalam PBO, objek digunakan untuk merepresentasikan hal-hal yang ada di dunia nyata atau konsep yang sesuai dengan aplikasi yang sedang dikembangkan.

Contoh :

```
class Mobil:
    def __init__(self, merek, model, tahun):
        self.merek = merek  # Atribut
        self.model = model  # Atribut
        self.tahun = tahun  # Atribut
        self.kecepatan = 0  # Atribut untuk menyimpan kecepatan mobil

    def akselerasi(self, increment):
        self.kecepatan += increment  # Metode untuk meningkatkan kecepatan
        print(f"Kecepatan mobil {self.merek} {self.model} sekarang adalah
{self.kecepatan} km/jam.")

    def rem(self, decrement):
        self.kecepatan -= decrement  # Metode untuk mengurangi kecepatan
        if self.kecepatan < 0:
            self.kecepatan = 0  # Kecepatan tidak bisa negatif
        print(f"Kecepatan mobil {self.merek} {self.model} sekarang adalah
{self.kecepatan} km/jam.")

# Membuat objek mobil1 dari kelas Mobil
mobil1 = Mobil("Toyota", "Camry", 2020)

# Menggunakan metode objek
mobil1.akselerasi(50)  # Meningkatkan kecepatan
mobil1.rem(20)         # Mengurangi kecepatan
```

5.Jelaskan jenis-jenis acces modifier beri contohnya dalam baris pemrograman?

Access modifier adalah dalam salah satu fingsi pemrograman berorientasi objek yang digunakan untuk menentukan tingkat aksesibilitas dari kelas, metode, dan atribut.

- Public
  Atribut dan metode yang dideklarasikan tanpa awalan tertentu dianggap sebagai public dan dapat diakses dari mana saja.
  Contoh :

```
class Mobil:
    def __init__(self, merek):
        self.merek = merek  # Atribut public
```

```
    def tampilkan_merek(self):  # Metode public
        print(f"Merek mobil: {self.merek}")

# Penggunaan
mobil1 = Mobil("Toyota")
mobil1.tampilkan_merek()  # Output: Merek mobil: Toyota
```

- Protected
  Atribut dan metode yang diawali dengan satu garis bawah (‘_’) dianggap sebagai protected. Ini menunjukkan bahwa mereka seharusnya tidak diakses dari luar kelas atau subclass.
  Contoh :

```
class Kendaraan:
    def __init__(self, tahun):
        self._tahun = tahun  # Atribut protected

    def _tampilkan_tahun(self):  # Metode protected
        print(f"Tahun produksi: {self._tahun}")

class Mobil(Kendaraan):
    def __init__(self, merek, tahun):
        super().__init__(tahun)
        self.merek = merek

    def tampilkan_info(self):
        print(f"Merek mobil: {self.merek}")
        self._tampilkan_tahun()  # Mengakses metode protected

# Penggunaan
mobil1 = Mobil("Toyota", 2020)
mobil1.tampilkan_info()
# Output:
# Merek mobil: Toyota
# Tahun produksi: 2020
```

- Private
  Atribut dan metode yang diawali dengan dua garis bawah (‘__’) dianggap sebagai private. Ini mencegah akses langsung dari luar kelas,sehingga nama atribut atau metode tersebut diubah di dalam kelas.
  Contoh :

```
class Mobil:
def __init__(self, merek):
self.__merek = merek  # Atribut private

def __tampilkan_merek(self):  # Metode private
print(f"Merek mobil: {self.__merek}")

def tampilkan_info(self):
```

```
self.__tampilkan_merek()  # Mengakses metode private dari dalam kelas

# Penggunaan
mobil1 = Mobil("Toyota")
mobil1.tampilkan_info()  # Output: Merek mobil: Toyota

# Akses langsung ke atribut private akan menyebabkan error
print(mobil1.__merek)
```

6.Gambarkan contoh pewarisan dalam diagram class?

```
class Kendaraan:
    def __init__(self, tahun):
        self.__tahunProduksi = tahun  # Atribut private

    def tampilkan_tahun(self):  # Metode public
        print(f"Tahun produksi: {self.__tahunProduksi}")

class Mobil(Kendaraan):
    def __init__(self, merek, tahun):
        super().__init__(tahun)  # Memanggil konstruktor superclass
        self.__merek = merek  # Atribut private

    def tampilkan_merek(self):  # Metode public
        print(f"Merek mobil: {self.__merek}")

    def tampilkan_info(self):  # Metode public
        self.tampilkan_merek()  # Mengakses metode dari subclass
        self.tampilkan_tahun()  # Mengakses metode dari superclass

# Penggunaan
mobil1 = Mobil("Toyota", 2020)
mobil1.tampilkan_info()
```