Pemrograman Berbasis Objek Praktik

Responsi 1

Maulana Rizal Alvani

5230411264

Sarjana Informatika

Universitas Teknologi Yogyakarta

1.Jelaskan perbedaan use case diagram dengan class diagram?

Use Case Diagram:

A.Fokus

Use case diagram berfokus pada interaksi antara pengguna (aktor) dengan sistem. Diagram ini menggambarkan bagaimana pengguna berinteraksi dengan sistem untuk mencapai tujuan tertentu.

B.Tujuan

Tujuan utama dari use case diagram adalah untuk memahami kebutuhan pengguna dan fungsionalitas sistem. Diagram ini membantu dalam mengidentifikasi aktor-aktor yang terlibat dalam sistem, menentukan use case-use case yang relevan, dan menggambarkan hubungan antara aktor dan use case.

C.Komponen utama Use Case Diagram:

-Aktor: Entitas yang berinteraksi dengan sistem. Aktor bisa berupa pengguna manusia, sistem eksternal, atau perangkat keras yang terhubung dengan sistem.

-Use Case: Fungsi atau tugas yang dilakukan oleh sistem dalam menanggapi aksi atau permintaan dari aktor. Use case menggambarkan fungsionalitas yang ditawarkan oleh sistem kepada aktor.

-Hubungan: Hubungan antara aktor dan use case digambarkan dengan menggunakan garis dan panah. Hubungan ini menunjukkan bagaimana aktor berinteraksi dengan use case dalam sistem.

-Penggunaan**:** Use case diagram digunakan untuk mengidentifikasi dan mengklasifikasikan kebutuhan fungsional sistem serta menggambarkan bagaimana pengguna berinteraksi dengan sistem. Diagram ini sangat berguna dalam tahap analisis dan desain awal pengembangan perangkat lunak.

Class diagram:

A.Fokus

Class diagram berfokus pada struktur kelas dan hubungan antara kelas-kelas dalam sistem. Diagram ini menggambarkan atribut dan metode yang dimiliki oleh setiap kelas serta hubungan antara kelas-kelas tersebut.

B.Tujuan

Tujuan utama dari class diagram adalah untuk memahami struktur sistem dan hubungan antara komponen-komponennya. Diagram ini membantu dalam merancang dan mendokumentasikan struktur sistem, termasuk atribut dan metode yang dimiliki oleh setiap kelas serta hubungan antara kelas-kelas tersebut.

C.Komponen utama Class Diagram

-Kelas: Entitas yang mewakili objek dalam sistem. Setiap kelas memiliki atribut (data) dan metode (fungsi) yang menggambarkan perilaku kelas tersebut.

-Atribut: Data yang dimiliki oleh kelas. Atribut menggambarkan karakteristik atau properti dari kelas.

-Metode: Fungsi yang dimiliki oleh kelas. Metode menggambarkan perilaku atau aksi yang dapat dilakukan oleh kelas.

-Hubungan: Hubungan antara kelas-kelas digambarkan dengan menggunakan garis dan panah. Hubungan ini menunjukkan bagaimana kelas-kelas saling berinteraksi dalam sistem.

-Penggunaan: Class diagram digunakan untuk merancang dan mendokumentasikan struktur sistem. Diagram ini sangat berguna dalam tahap desain dan implementasi pengembangan perangkat lunak, karena membantu dalam memahami bagaimana komponen-komponen sistem saling berhubungan dan berinteraksi.

2.Jelaskan jenis-jenis dependensi

A. Association

Association adalah hubungan antara dua kelas yang menunjukkan bahwa objek dari satu kelas dapat berinteraksi dengan objek dari kelas lain.

B. Aggregation

Aggregation adalah jenis asosiasi khusus yang menunjukkan hubungan "bagian-dari" antara dua kelas. Dalam hubungan ini, satu kelas (komponen) adalah bagian dari kelas lain (keseluruhan), tetapi komponen tersebut dapat eksis secara independen dari keseluruhan.

C. Composition

Composition adalah jenis agregasi yang lebih kuat, di mana komponen tidak dapat eksis secara independen dari keseluruhan. Dalam hubungan ini, jika keseluruhan dihapus, maka komponen juga akan dihapus.

D. Dependency

Dependency adalah hubungan di mana satu kelas bergantung pada kelas lain untuk berfungsi. Ini menunjukkan bahwa perubahan pada satu kelas dapat mempengaruhi kelas lain.

E.Generalization

Generalization adalah hubungan hierarkis antara kelas induk dan kelas anak, di mana kelas anak mewarisi atribut dan metode dari kelas induk. Ini menunjukkan hubungan "is-a" antara kelas.

F.Realization

Realization adalah hubungan antara antarmuka dan kelas yang mengimplementasikan antarmuka tersebut. Ini menunjukkan bahwa kelas tersebut menyediakan implementasi konkret dari metode yang didefinisikan dalam antarmuka.

3. Apa perbedaan pemrograman terstruktur dengan berorientasi objek, jelaskan

-Pendekatan:

Pemrograman terstruktur menggunakan pendekatan prosedural dengan fokus pada alur logika program, sedangkan **OOP** menggunakan pendekatan berbasis objek dengan fokus pada objek dan interaksi antar objek.

-**Modularitas:**

Pemrograman terstruktur memecah program menjadi fungsi-fungsi, sedangkan **OOP** memecah program menjadi objek dan kelas.

-**Pengelolaan Data**:

Dalam **pemrograman terstruktur**, data dan fungsi dipisahkan, sedangkan dalam **OOP**, data dan fungsi digabungkan dalam objek.

-**Pewarisan**:

OOP mendukung pewarisan sifat dan perilaku melalui inheritance, sedangkan **pemrograman terstruktur** tidak memiliki konsep pewarisan.

-Bahasa:

Pemrograman Struktur menggunakan contoh bahasa C, Pascal, Fortran. Sedangkan **OOP** menggunakan conroh bahasa Java, C++, Python, Ruby.

4. Jelaskan konsep objek dan beri contohnya?

Objek yaitu entitas yang memiliki atribut (data) dan metode (fungsi) yang menggambarkan perilaku atau aksi yang dapat dilakukan oleh objek tersebut. Objek adalah instansiasi dari kelas, yang merupakan cetak biru atau template untuk membuat objek.

Objek memiliki karakteristik utama seperti:

-Atribut: Data atau properti yang dimiliki oleh objek. Atribut menggambarkan karakteristik atau keadaan objek.

-Metode: Fungsi atau prosedur yang dapat dilakukan oleh objek. Metode menggambarkan perilaku atau aksi yang dapat dilakukan oleh objek.

-Encapsulation: Penggabungan data (atribut) dan fungsi (metode) dalam satu unit yang disebut objek. Encapsulation membantu melindungi data dari akses langsung dan hanya dapat diakses melalui metode yang disediakan.

-Identity: Setiap objek memiliki identitas unik yang membedakannya dari objek lain, bahkan jika mereka memiliki atribut yang sama.

Contoh

contoh4.py

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

    def jalan(self):
        print(f"{self.merk} sedang berjalan dengan kecepatan {self.kecepatan} km/jam")

    def berhenti(self):
        print(f"{self.merk} telah berhenti")

    def belok(self, arah):
        print(f"{self.merk} belok ke {arah}")

# Membuat objek dari kelas Mobil
mobil1 = Mobil("Merah", "Toyota", 60)
mobil2 = Mobil("Biru", "Honda", 80)

# Menggunakan metode pada objek
mobil1.jalan()  # Output: Toyota sedang berjalan dengan kecepatan 60 km/jam
mobil2.berhenti()  # Output: Honda telah berhenti
mobil1.belok("kiri")  # Output: Toyota belok ke kiri
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

PS C:\Users\ASUS\OneDrive\Desktop\PBOP> & C:/Users/ASUS/AppData/Local/Microsoft/WindowsApps/python3.11.exe
Toyota sedang berjalan dengan kecepatan 60 km/jam
Honda telah berhenti
Toyota belok ke kiri
```

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

A. Public

Anggota dengan access modifier public bisa diakses dari mana saja, baik dari dalam kelas yang sama, kelas lain dalam paket yang sama, maupun kelas di paket berbeda. Modifier ini memberikan visibilitas penuh kepada anggota kelas.

Contoh:

```
class Mobil:
    def __init__(self, merk):
        self.merk = merk  # Public attribute

    def jalan(self):
        print(f"{self.merk} sedang berjalan.")  # Public method

# Membuat objek dari kelas Mobil
mobil1 = Mobil("Toyota")
mobil1.jalan()  # Output: Toyota sedang berjalan.
```

B.Private

Anggota kelas yang dideklarasikan sebagai private hanya dapat diakses dari dalam kelas itu sendiri. Anggota ini tidak dapat diakses dari kelas lain, bahkan dari kelas turunan.

Contoh:

```
class Mobil:
    def __init__(self, merk):
        self.__merk = merk  # Private attribute

    def set_merk(self, merk):
        self.__merk = merk  # Public method to set private attribute

    def jalan(self):
        print(f"{self.__merk} sedang berjalan.")  # Public method

# Membuat objek dari kelas Mobil
mobil1 = Mobil("Toyota")
mobil1.jalan()  # Output: Toyota sedang berjalan.
# mobil1.__merk  # Akan menghasilkan error karena __merk adalah atribut
private
```

C.Protected

Anggota kelas yang dideklarasikan sebagai protected dapat diakses dari dalam kelas itu sendiri, dari kelas lain dalam paket yang sama, dan dari kelas turunan (subclass), baik dalam paket yang sama maupun berbeda.

Contoh:

```
class Mobil:
    def __init__(self, merk):
        self._merk = merk  # Protected attribute

    def jalan(self):
        print(f"{self._merk} sedang berjalan.")  # Public method

class MobilBalap(Mobil):
```

```
    def tampilkan_merk(self):
        print(f"Merk mobil balap: {self._merk}")  # Accessing protected
attribute

# Membuat objek dari kelas MobilBalap
mobil_balap = MobilBalap("Ferrari")
mobil_balap.tampilkan_merk()  # Output: Merk mobil balap: Ferrari
```

D.Default (Package-Private)

Dalam Python, tidak ada access modifier default seperti di Java. Namun, atribut yang tidak diawali dengan underscore dianggap sebagai public.

Contoh:

```
class Mobil:
    def __init__(self, merk):
        self.merk = merk  # Default (public) attribute

    def jalan(self):
        print(f"{self.merk} sedang berjalan.")  # Public method

# Membuat objek dari kelas Mobil
mobil1 = Mobil("Toyota")
mobil1.jalan()  # Output: Toyota sedang berjalan.
```

6. Gambarkan contoh pewarisan dalam diagram class?