Pemrograman Berbasis Objek Praktik

Responsi 1

Nama : Zidan Amikul Afham

NPM : 5230411330

Sarjana Informatika

Universitas Teknologi Yogyakarta

Tahun ajaran 2023/2024

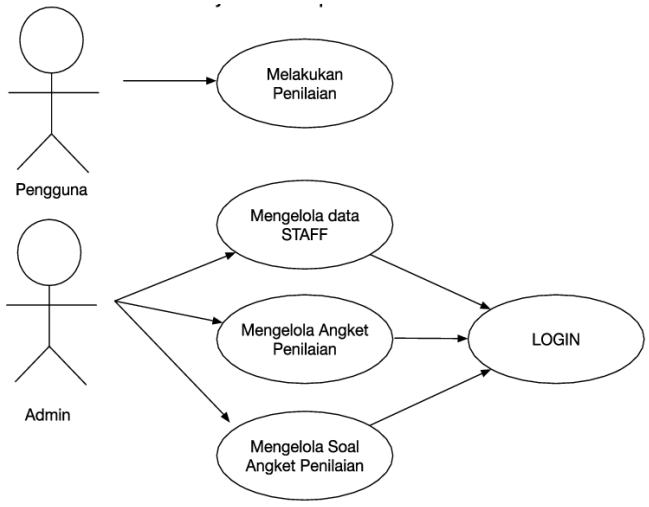
Responsi

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 access modifier beri contohnya dalam baris pemrograman?
6. Gambarkan contoh pewarisan dalam diagram class?

1. Perbedaan Use Case Diagram dan Class Diagram

- **Use Case Diagram**: Diagram ini digunakan untuk menggambarkan bagaimana aktor (pengguna atau sistem eksternal) berinteraksi dengan sistem yang sedang dikembangkan. Use Case Diagram membantu dalam memahami kebutuhan fungsional sistem dengan menampilkan aktivitas atau fungsionalitas yang bisa dijalankan oleh pengguna. Diagram ini terdiri dari aktor (orang atau sistem yang menggunakan aplikasi) dan use case (aktivitas atau fungsi yang dapat dilakukan sistem). Setiap use case menunjukkan interaksi antara pengguna dengan sistem dan membantu tim pengembang memahami apa saja yang perlu dilakukan sistem agar memenuhi kebutuhan pengguna.

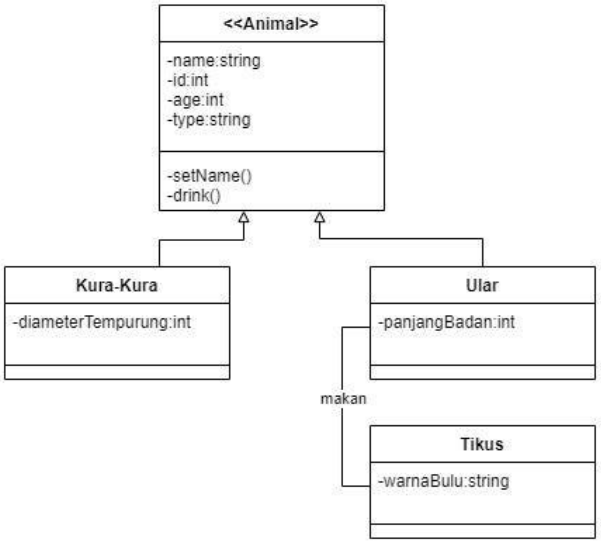
  Contoh :



- **Class Diagram**: Class diagram adalah diagram yang menunjukkan struktur sistem dengan menggambarkan kelas-kelas, atribut, metode, dan hubungan antar kelas dalam sebuah sistem. Diagram ini menggambarkan bagaimana data dan fungsi terstruktur di

dalam sistem. Setiap kelas biasanya mewakili entitas atau objek dalam sistem, seperti "Pengguna" atau "Produk" dalam sistem e-commerce. Kelas-kelas ini memiliki atribut (data atau karakteristik yang dimiliki kelas) dan metode (fungsi atau operasi yang dapat dilakukan kelas). Class diagram juga menunjukkan berbagai jenis hubungan antar kelas, seperti asosiasi, agregasi, komposisi, dan pewarisan. Hal ini membantu pengembang memahami struktur dan bagaimana bagian-bagian sistem saling terhubung satu sama lain.

Contoh :



2. Jenis jenis dependensi

Dalam pemrograman berbasis objek, terdapat beberapa jenis dependensi yang umum, yaitu:

1. Association: Merupakan hubungan antar kelas di mana satu kelas menggunakan atau merujuk ke kelas lain. Hubungan ini bisa bersifat *unidirectional* (satu kelas mengetahui kelas lain) atau *bidirectional* (kedua kelas saling mengetahui). Ini menunjukkan adanya hubungan antar objek tanpa keterikatan kuat.

2. Aggregation: Bentuk *association* yang lebih khusus, di mana satu objek memiliki objek lain, tetapi keduanya dapat berdiri sendiri. Hubungan ini dikenal sebagai *has-a*, di mana objek yang dimiliki tetap dapat eksis meski objek induknya dihapus, menunjukkan hubungan yang relatif longgar.

3. Composition: Hubungan yang lebih kuat dari *aggregation*, di mana objek yang dimiliki sepenuhnya bergantung pada objek induknya. Jika objek induk dihancurkan,

objek yang menjadi bagiannya juga ikut dihancurkan. Hubungan ini menunjukkan ketergantungan penuh dan disebut *part-of*.

4. Dependency Injection: Teknik untuk mengurangi keterikatan antarkelas dengan memberikan objek yang dibutuhkan ke dalam kelas dari luar, melalui konstruktor, metode, atau properti. Teknik ini membantu membuat kode lebih modular dan mudah diuji dengan menghindari pembuatan objek di dalam kelas.

5. Inheritance (Pewarisan): Hubungan di mana satu kelas (subclass) mewarisi atribut dan metode dari kelas induk (superclass). Ini memungkinkan penggunaan kembali kode, perluasan fungsionalitas, dan mendukung polimorfisme, di mana kelas anak dapat memodifikasi atau menambahkan perilaku baru.

6. General Dependency: Hubungan di mana satu kelas menggunakan metode atau objek dari kelas lain untuk menyelesaikan tugas tertentu. Dependensi ini menunjukkan bahwa satu kelas memerlukan kelas lain, misalnya, melalui penggunaan parameter metode atau variabel lokal.

3. Perbedaan Pemrograman Terstruktur dengan Pemrograman Berorientasi Objek (OOP)

- **Pemrograman Terstruktur**: Pemrograman terstruktur menggunakan pendekatan prosedural atau berorientasi pada prosedur, di mana kode dipecah ke dalam fungsi atau prosedur yang lebih kecil. Setiap fungsi ini menjalankan tugas tertentu sesuai dengan urutan alur program. Biasanya digunakan untuk program yang lebih sederhana, pemrograman terstruktur memiliki struktur yang linear dan terkendali, sehingga mudah diikuti dan dipahami. Kelemahannya adalah ketika program menjadi lebih besar, sulit untuk dikelola, terutama jika banyak fungsi yang saling bergantung.
- **Pemrograman Berorientasi Objek (OOP)**: OOP menggunakan pendekatan berbasis objek, di mana program dipecah ke dalam objek-objek yang masing-masing mewakili entitas dunia nyata. Objek-objek ini menggabungkan data (disebut atribut) dan perilaku (disebut metode) yang dimiliki suatu entitas. OOP memungkinkan modularitas lebih baik dengan konsep-konsep utama seperti:
  - **Encapsulation**: Menyembunyikan detail data di dalam objek, sehingga hanya metode tertentu yang bisa mengakses atau mengubah data.
  - **Inheritance**: Membuat kelas baru yang mewarisi sifat-sifat dari kelas lain, sehingga kode menjadi lebih reusable.

- **Polymorphism**: Memungkinkan objek untuk memiliki bentuk yang berbeda tergantung pada konteksnya, meningkatkan fleksibilitas dalam penggunaan metode.
- **Abstraction**: Menyederhanakan kompleksitas dengan fokus hanya pada detail penting, menyembunyikan detail yang tidak relevan untuk pengguna. Dengan OOP, pengembangan sistem besar jadi lebih terstruktur, modular, dan mudah dikelola dalam jangka panjang.

4. Konsep Objek dalam Pemrograman Berorientasi Objek (OOP)

Dalam pemrograman berorientasi objek, **objek** adalah “benda” atau “unit” yang memiliki data dan kemampuan. Kita menggunakan objek untuk mewakili hal-hal dari dunia nyata, seperti mobil, hewan, orang, dan sebagainya. Setiap objek punya data dan bisa melakukan tindakan tertentu.

Objek memiliki dua elemen utama:

1.Atribut (Properties/Attributes): Atribut adalah data atau karakteristik yang dimiliki objek. Atribut mencerminkan informasi yang melekat pada objek tersebut.

2.Metode (Methods/Behavior): Metode adalah fungsi atau tindakan yang bisa dilakukan oleh objek. Metode ini memungkinkan objek untuk berinteraksi atau mengubah nilai atributnya.

Contoh dalam Kehidupan Nyata

Misalnya, objek **"Mobil"**. Sebuah mobil memiliki berbagai karakteristik dan perilaku sebagai berikut:

- **Atribut**:
  - **Warna**: Menggambarkan warna mobil, misalnya merah.
  - **Merk**: Menyatakan merk mobil, seperti Toyota atau Honda.
  - **Kecepatan**: Menunjukkan kecepatan mobil dalam km/jam.
- **Metode**:
  - **Maju()**: Fungsi yang membuat mobil bergerak maju.
  - **Mundur()**: Fungsi yang membuat mobil bergerak mundur.
  - **Rem()**: Fungsi untuk mengurangi atau menghentikan kecepatan.

Objek "Mobil" dibuat berdasarkan blueprint (cetakan) yang disebut **kelas "Mobil"**. Saat kelas ini diinstansiasi, terbentuklah objek yang nyata, misalnya mobil dengan warna merah, merk Toyota, dan kecepatan 60 km/jam.

5. Jenis jenis access modifier dan contohnya

1. Public

- **Pengertian**: Atribut atau metode yang bersifat public dapat diakses dari mana saja, baik di dalam kelas itu sendiri maupun dari luar kelas.
- **Ket** : Tidak ada tanda khusus yang digunakan,semua atribut dan metode secara default adalah public.

  Contoh :

```python
class Example:
    def __init__(self):
        self.public_var = 10

    def public_method(self):
        print("This is a public method")

obj = Example()
print(obj.public_var)
obj.public_method()
```

2. Protected

- **Pengertian**: Atribut atau metode yang bersifat protected sebaiknya hanya diakses di dalam kelas itu sendiri dan subclass. Namun, ini hanya konvensi dan Python tidak memberikan pembatasan ketat.
- **Ket** : Ditandai dengan satu underscore (_).

  Contoh:

```python
class Example:
    def __init__(self):
        self._protected_var = 20

    def _protected_method(self):
        print("This is a protected method")

obj = Example()
print(obj._protected_var)
obj._protected_method()
```

3. Private

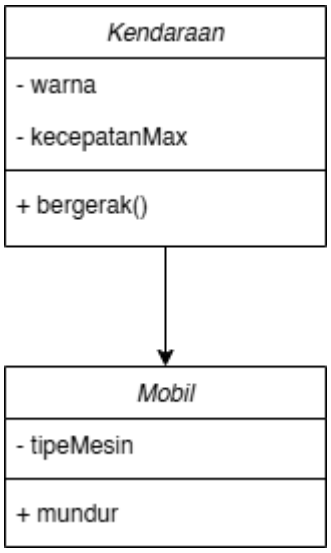
- **Pengertian**: Atribut atau metode private hanya bisa diakses di dalam kelas itu sendiri. Python menggunakan *name mangling* untuk mengubah nama atribut agar tidak mudah diakses dari luar.
- **Ket** : Ditandai dengan dua underscore di awal (__).

  Contoh:

```python
class Example:
    def __init__(self):
        self.__private_var = 30

    def __private_method(self):
        print("ini privat")

    def access_private_method(self):
        self.__private_method()

obj = Example()

print(obj._Example__private_var)
obj._Example__private_method()
```

6. Gambar Contoh Pewarisan

Saya gambar menggunakan draw.io



Pada diagram di atas, Kendaraan adalah superclass yang mewakili kendaraan secara umum dengan atribut warna dan kecepatanMax, serta metode bergerak(). Sedangkan Mobil adalah subclass atau turunan dari Kendaraan, yang berarti Mobil mewarisi atribut warna dan kecepatanMax, serta metode bergerak() dari Kendaraan. Kelas Mobil juga memiliki atribut tambahan yaitu tipeMesin dan metode mundur() yang spesifik hanya untuk kelas Mobil.