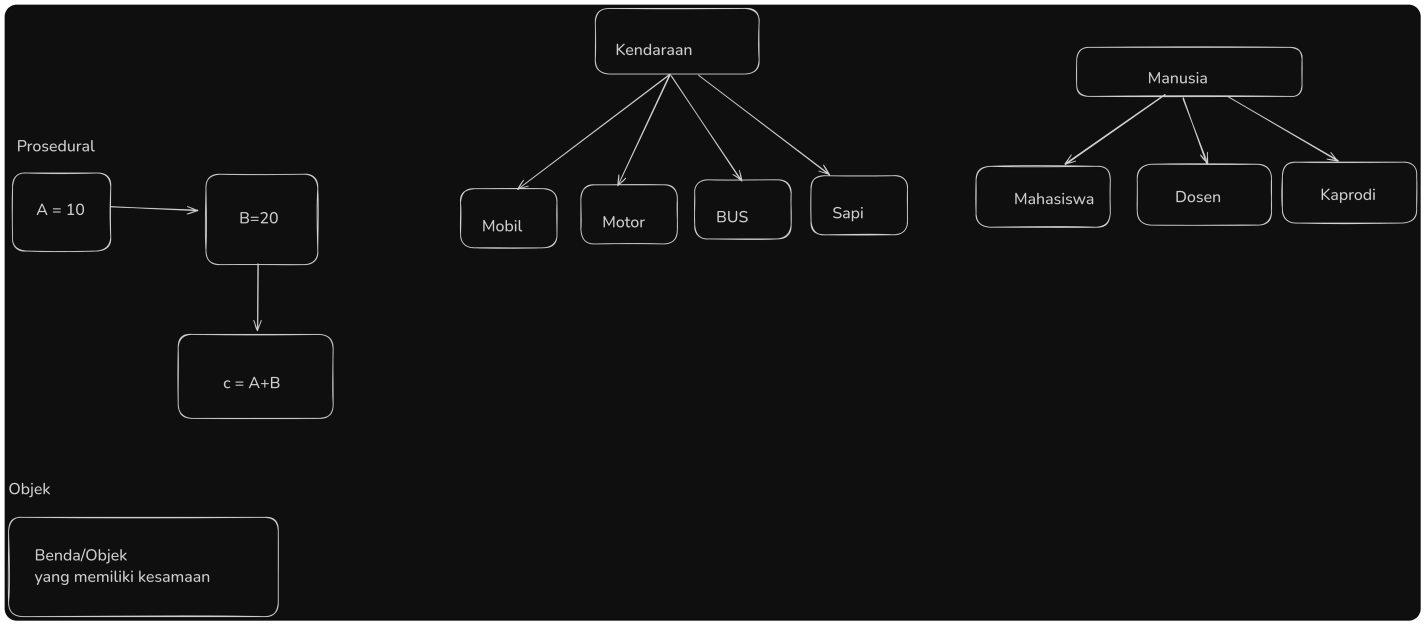
Penjelasan tentang kelas dan objek



1. Class

Class adalah blueprint atau cetak biru yang mendefinisikan atribut (properti) dan perilaku (metode) dari sekelompok objek. Class tidak berisi data spesifik; sebaliknya, class mendefinisikan apa yang dimiliki oleh setiap objek yang dibuat berdasarkan class tersebut.

Misalnya, jika kita memiliki class `Mobil`, maka class ini bisa memiliki atribut seperti `warna`, `merk`, dan `model`, serta metode seperti `jalan()` atau `berhenti()`.

Contoh di Python:

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

    def jalan(self):
        print(f"{self.merk} sedang berjalan.")
```

Dalam contoh di atas:

- `__init__` adalah metode khusus yang disebut saat objek dibuat, untuk menginisialisasi properti.
- `merk` dan `warna` adalah atribut yang akan dimiliki setiap objek `Mobil`.
- `jalan()` adalah metode yang mendefinisikan perilaku objek.

2. Object

Object adalah instansiasi dari class, atau dengan kata lain, objek adalah hasil nyata dari class yang dapat digunakan. Setiap

objek memiliki salinan atribut dan metode yang didefinisikan oleh class tersebut, tetapi dengan nilai yang bisa berbeda-beda.

Contoh membuat objek dari class `Mobil`:

```
mobil_saya = Mobil("Toyota", "Merah", "DR 0191 XY")
mobil_budi = Mobil("Honda", "Hitam", "DR 0189 XY")
mobil_saya.jalan()  # Output: Toyota sedang berjalan.
```

Di sini, `mobil_saya` adalah objek dari class `Mobil`. Ia memiliki atribut `merk` yang nilainya "Toyota" dan `warna` yang nilainya "Merah". Objek tersebut bisa menggunakan metode `jalan()` untuk mengeksekusi perilaku yang sudah didefinisikan dalam class.

Hubungan Class dan Object:

- Class adalah konsep abstrak, sementara object adalah realisasi konkret dari class.
- Class seperti cetak biru dari sebuah rumah, sedangkan object adalah rumah yang dibangun berdasarkan cetak biru tersebut.

Dengan menggunakan class, kita dapat membuat banyak objek dengan properti dan perilaku yang serupa, tetapi dengan data yang berbeda.

Berikut adalah contoh lain dari class dan object dalam pemrograman berorientasi objek. Kali ini kita akan membuat class **Mahasiswa** yang menggambarkan atribut dan metode untuk objek mahasiswa.

Contoh Class `Mahasiswa`

```
class Mahasiswa:
    def __init__(self, nama, nim, jurusan):
        self.nama = nama      # Atribut nama
        self.nim = nim        # Atribut nim
        self.jurusan = jurusan # Atribut jurusan

    def belajar(self, mata_kuliah):
        print(f"{self.nama} sedang belajar {mata_kuliah}.")

    def info(self):
        print(f"Nama: {self.nama}, NIM: {self.nim}, Jurusan: {self.jurusan}")
```

Penjelasan:

- `__init__`: Konstruktor yang digunakan untuk menginisialisasi objek dengan atribut `nama`, `nim`, dan `jurusan`.
- `belajar()`: Metode yang mencetak pesan ketika mahasiswa sedang belajar suatu mata kuliah.
- `info()`: Metode untuk menampilkan informasi lengkap mahasiswa.

Contoh Penggunaan:

```
# Membuat objek Mahasiswa
mahasiswa1 = Mahasiswa("Andi", "123456", "Teknik Informatika")
mahasiswa2 = Mahasiswa("Budi", "789012", "Sistem Informasi")

# Memanggil metode belajar
mahasiswa1.belajar("Pemrograman Berorientasi Objek")  # Output: Andi sedang belajar Pemrograman
Berorientasi Objek.
mahasiswa2.belajar("Basis Data")  # Output: Budi sedang belajar Basis Data.

# Memanggil metode info
mahasiswa1.info()  # Output: Nama: Andi, NIM: 123456, Jurusan: Teknik Informatika
mahasiswa2.info()  # Output: Nama: Budi, NIM: 789012, Jurusan: Sistem Informasi
```

Hasil:

- Objek `mahasiswa1` dan `mahasiswa2` adalah instansiasi dari class `Mahasiswa` dengan data yang berbeda.
- Kedua objek tersebut dapat memanggil metode yang sama, tetapi menampilkan hasil sesuai dengan atribut masing-masing.

Dengan menggunakan class seperti ini, kita bisa merepresentasikan mahasiswa dengan atribut dan perilakunya di dalam kode.