Nama : Gita Sari

Nim : 236270035

Jurusan : Sistem Informasi

Pengertian OOP

Pemrograman Berorientasi Objek (OOP) adalah suatu paradigma pemrograman yang menggunakan "objek" dan "kelas" untuk merancang aplikasi dan program. OOP berfokus pada cara untuk mengorganisasi data dan perilaku dalam bentuk objek yang saling berinteraksi.

Berikut adalah beberapa konsep utama dalam OOP:

1. Kelas (Class)**:** Merupakan cetak biru atau blueprint untuk membuat objek. Kelas mendefinisikan atribut (data) dan metode (fungsi) yang dimiliki oleh objek. Sebagai contoh, kelas "Mobil" bisa memiliki atribut seperti "warna" dan "merk", serta metode seperti "jalan" atau "berhenti".
2. Objek (Object)**:** Merupakan instansi (salinan) dari sebuah kelas. Setiap objek memiliki keadaan (nilai dari atribut) dan perilaku (fungsi atau metode) yang ditentukan oleh kelas. Misalnya, objek "Mobil1" adalah instansi dari kelas "Mobil".
3. Enkapsulasi (Encapsulation)**:** Proses menyembunyikan detail implementasi dalam kelas dan hanya memperlihatkan antarmuka yang dibutuhkan oleh pengguna kelas tersebut. Hal ini memungkinkan pengelolaan data yang lebih aman dan mudah.
4. Pewarisan (Inheritance)**:** Konsep di mana sebuah kelas (kelas anak) dapat mewarisi atribut dan metode dari kelas lain (kelas induk). Ini memungkinkan penggunaan kembali kode dan pengorganisasian hierarki yang lebih baik. Misalnya, kelas "Mobil" dapat menjadi kelas induk, sementara kelas "Sedan" dan "SUV" adalah kelas turunan.
5. Polimorfisme (Polymorphism)**:** Kemampuan objek untuk mengambil berbagai bentuk. Dalam OOP, polimorfisme berarti metode yang sama dapat digunakan pada objek yang berbeda, dan setiap objek dapat memberikan implementasi yang berbeda untuk metode tersebut. Misalnya, metode "suara()" pada objek "Anjing" bisa menghasilkan suara "Guk-guk", sedangkan pada objek "Kucing" bisa menghasilkan suara "Meong".
6. Abstraksi (Abstraction)**:** Proses menyembunyikan kompleksitas dan hanya menunjukkan fitur yang relevan kepada pengguna. Dalam OOP, ini sering kali dilakukan dengan menggunakan kelas abstrak atau antarmuka (interface) yang mendefinisikan metode yang harus diimplementasikan oleh kelas turunan.

1. Struktur Kelas

```
class Namakelas:
    def __init__(self):
```

```
        self.__dataPribadi = "ini adalah variabel Private"
```

Konstruktor (__init__):

Fungsi ini dihasilkan saat objek dari kelas Namakelas dibuat. Di sini, __dataPribadi diinisialisasi sebagai variabel private, yang hanya dapat diakses dari dalam kelas itu sendiri. Penggunaan dua garis bawah (__) menandakan bahwa ini adalah variabel yang harus disembunyikan dari akses luar.

2. Metode Private

```
    def __metodePribadi(self):
        return "Ini adalah metode Private"
```

Metode Private:

__metodePribadi adalah metode yang tidak dapat dipanggil dari luar kelas. Hal ini melindungi logika yang mungkin sensitif atau tidak ingin diekspos. Metode ini hanya bisa diakses oleh metode lain dalam kelas yang sama.

3. Getter dan Setter

```
    def getDataPribadi(self):
        return self.__dataPribadi


    def setDataPribadi(self, nilaiBaru):
        self.__dataPribadi = nilaiBaru
```

Getter:

getDataPribadi memungkinkan akses kepada nilai __dataPribadi tanpa memberikan akses langsung. Ini merupakan bagian dari prinsip enkapsulasi yang menjaga integritas data.

Setter:

setDataPribadi memungkinkan pengguna untuk mengubah nilai __dataPribadi. Ini juga mengontrol bagaimana nilai dapat diubah, menjaga validitas data jika perlu.

4. Metode Akses

```
    def aksesMetodePribadi(self):
        return self.__metodePribadi()
```

Akses Metode Private:

aksesMetodePribadi berfungsi sebagai perantara untuk memanggil metode private. Ini menunjukkan bagaimana metode publik dalam kelas dapat mengakses metode private, yang tidak bisa diakses langsung dari luar.

5. Metode Publik

```
def metodePublik(self):
    print("Ini adalah metode publik")
    print(self.__metodePribadi())
    print(self.__dataPribadi)
```

Metode Publik:

metodePublik dapat dipanggil dari luar kelas dan berfungsi untuk memberikan informasi tentang keadaan objek. Dalam metode ini, meskipun __metodePribadi dan __dataPribadi adalah private, mereka dapat diakses karena berada di dalam konteks kelas yang sama.

Kelas TurunanNamaKelas

1. Pewarisan

```
class TurunanNamaKelas(Namakelas):
    def __init__(self):
        super().__init__()
```

Inheritance:

Kelas ini mewarisi semua atribut dan metode dari Namakelas. Dengan menggunakan super().__init__(), konstruktor dari kelas induk dipanggil, sehingga atribut private juga diinisialisasi. Ini memastikan bahwa semua data awal dari kelas induk tetap konsisten.

2. Metode Turunan

```
    def demoTurunan(self):
        dataPribadi = self.getDataPribadi()
        print(f"Data pribadi dari kelas induk: {dataPribadi}")

        self.setDataPribadi("nilai baru dari turunan")
        dataPribadiBaru = self.getDataPribadi()
        print(f"Data pribadi setelah diubah oleh turunan: {dataPribadiBaru}")

        hasilMetodePribadi = self.aksesMetodePribadi()
        print(f"Hasil akses metode pribadi dari turunan: {hasilMetodePribadi}")
```

Demonstrasi Akses:

Di dalam metode demoTurunan, kelas ini menggunakan getter untuk mengambil __dataPribadi, menunjukkan akses yang aman terhadap data private dari kelas induk.

Setter digunakan untuk memperbarui nilai __dataPribadi. Ini menunjukkan fleksibilitas dalam memanipulasi data private melalui kontrol yang ada pada kelas induk.

Dengan menggunakan aksesMetodePribadi, kelas turunan dapat memanggil metode private, menunjukkan bagaimana meskipun metode ini tersembunyi dari luar, mereka masih dapat diakses dalam hierarki kelas.

Kesimpulan

Melalui penerapan encapsulation dan inheritance, kode ini mengedepankan prinsip OOP yang baik:

Encapsulation: Mengamankan data dengan menyembunyikannya dan memberikan akses melalui metode. Ini menjaga integritas data dan memberikan kontrol penuh terhadap bagaimana data diakses dan dimodifikasi.

Inheritance: Memungkinkan penggunaan kembali kode dan memperluas fungsionalitas. Kelas turunan dapat mengakses dan mengubah perilaku kelas induk tanpa perlu menduplikasi kode.