LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

Kelas Abstrak, Interface, dan Metaclass

Disusun Oleh :

Nama : Rizki Esa Fadillah

NIM : 121140084

Kelas : PBO – RB

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2023

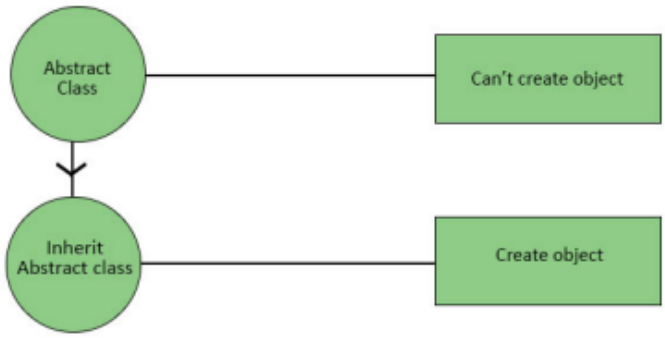
RINGKASAN

A. Abstrak

Dalam pengembangan program komputer semakin besar program yang dibuat maka akan semakin kompleks pula baris kode yang dibuat. Oleh karena itu, perlu adanya konsep untuk mengurangi kompleksitas salah satunya menggunakan konsep OOP (Objek Oriented Programing) abstraksi. Konsep abstraksi sendiri dapat mengurangi kompleksitas dengan cara menyembunyikan detail – detail yang penting dari pengguna (user) dan hanya menampilkan atribut yang esensial.

Kelas Abstrak merupakan sebuah kelas yang didalamnya terdapat satu atau lebih metode abstrak. Metode abstrak adalah metode yang ter deklarasi namun tidak ada implementasinya. Dengan menggunakan kelas abstrak, dapat menetukan Antarmuka Program Aplikasi (API) umum untuk satu sebkelas. Sehingga akan membantu saat bekerja dengan basis kode besar dimana menyimpan semua keals.

Kelas abstrak digunakan didalam inheritance class atau Override methode untuk membuat struktur logika penurunan di dalam pemrograman objek.



- **Implementasi Kelas Abstrak**

  Kelas abstrak dapat diimplementasikan menggunakan modul ABC. ABC bekerja dengan metode atribut class dasar sebagai abstrak dan kemudian mendaftarkan kelas beton (Berisi metode konkret/normal) sebagai implementasi dari basis abstrak. Kelas dasar abstrak juga dapat menyediakan implementasi dengan memanggil konstruktor metode class parent melalui super(). Metode menjadi abstrak dengan menggunakan kata kunci @abstracmethod.

```python
from abc import ABC, abstractmethod # implemntasi modul ABC
class Bangun_Datar(ABC):
    @abstractmethod
    def Jumlah_sisi(self):
        pass
class Segitiga(Bangun_Datar):
    # overriding abstract method
    def Jumlah_sisi(self):
        print("Segitiga memiliki 3 sisi")
class Lingkaran(Bangun_Datar):
    # overriding abstract method
    def Jumlah_sisi(self):
        print("Lingkaran memiliki 1 sisi")

R = Segitiga()
R.Jumlah_sisi()
R = Lingkaran()
R.Jumlah_sisi()
```

Output :

```
PS D:\Belajar Python> & C:/Users/
Segitiga memiliki 3 sisi
Lingkaran memiliki 1 sisi
PS D:\Belajar Python>
```

B. Interface

Interface merupakan koleksi dari method/ fungsi yang perlu disediakan oleh implementing class (child class). Pada tingakat yang lebih tinggi, interface bekerja sebagai cetak biru untuk mendesain tampilan class. Interface memiliki metodde yang bersifat abstract. Metode abstract akan memiliki satu-satunya deklarasi karena tidak adanya implementasi. Pengimplementasian sebuah interface adalah sebuah cara untuk menulis kode yang elegan dan terorganisir.

Perbedaan Interface dan Abstrak

Abstrak :

1. Memiliki access specifier.
2. Dapat berisi implementasi secara lengkap.
3. Kecepatan proses cepat.
4. Berisi field dan konstanta
5. Terdapat method abstrak dan konkret

Interface :

1. Tidak memiliki access spesifer.
2. Class tidak terdapat implementasi.
3. Kecepatan proses lambat

4. Tidak dapat berisi field.
5. Hanya terdapat method abstrak.

- Informal Interface

  Implementasi interface di python kadang diartikan sebagia konsep tanpa aturan yang terlalu ketat. Interface informal python adalah kelas yang mendefinisikan metode yang dapat diganti, tetapi penerapannya tidak ketat. Sehingga untuk mengimplementasikannya membutuhkan class konkrit. Class konkrit adalah subclass dari abstrak interface yang menyediakan implementasi dari method/fungsi di kelas interface.

```
1  class Masakan:
2      def __init__(self, menu):
3          self.__daftar_menu = menu
4      def __len__(self):
5          return len(self.__daftar_menu)
6      def __contains__(self, menu):
7          return menu in self.__daftar_menu
8
9  class Restoran(Masakan):
10     def __iter__(self):
11         return iter(self._Masakan__daftar_menu)
12
13 Masakan_Padang = Restoran(["Rendang", "Gulai", "Dendeng", "Tambusu"])
14 print(len(Masakan_Padang))
15 print("Rendang" in Masakan_Padang)
16 print("Opor" in Masakan_Padang)
17 print("Daftar menu Restoran Masakan Padang : ")
18 for Masakan in Masakan_Padang:
19     print(Masakan)
```

Output :

```
4
True
False
Daftar menu Restoran Masakan Padang :
Rendang
Gulai
Dendeng
Tambusu
PS D:\Belajar Python>
```

- Formal Interface

  Formal Interface dapat berguna untuk proyek dengan basis kode kecil dan sejumlah pemrogram. Pada Formal interface kelas parent (abstrak) dapat dibangun dengan sedikit baris kode, untuk diimplementasikan pada kelas turunannya (konkret). Implementasinya bersifat wajib (formal), untuk membuat formal interface dierlukan beberapa alat dari abc modul Python.

```python
from abc import ABC, abstractmethod

class Musik(ABC):
    @abstractmethod
    def Nyalakan_Musik(self):
        pass
    def Matikan_Musik(self):
        pass

class Handphone(Musik):
    def Nyalakan_Musik(self):
        print("Musik dinyalakan")
    def Matikan_Musik(self):
        print("Musik dimatikan")

Vivo = Handphone()

Vivo.Nyalakan_Musik()
```

Output :

```
PS D:\Belajar Python> & C:/Use
Musik dinyalakan
PS D:\Belajar Python>
```

C. Metaclass

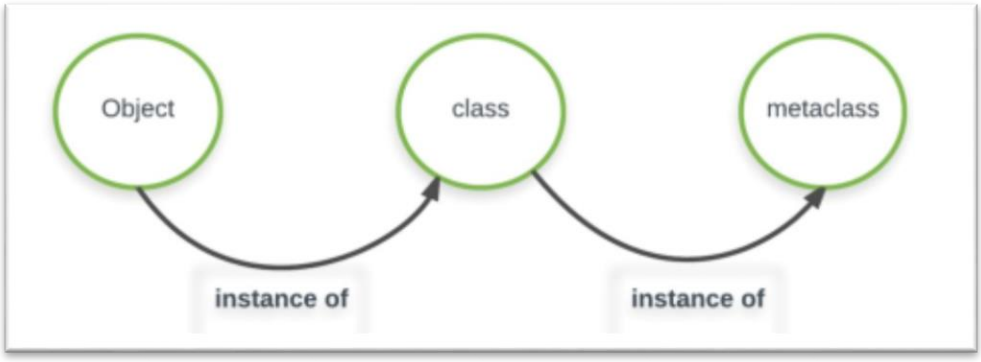
Metaclass di Python adalah kelas dari kelas yang mendefinisikan bagaimana kelas berperilaku. pada Python semua tipe pada dasarnya adalah suatu objek/kelas juga (termasuk int, float, dll). Dengan kata lain, metaclass adalah tipe kelas spesial yang berfungsi untuk membuat kelas-kelas pada Python (atau biasa disebut juga dengan pabrik kelas/class factory). Kelas dalam Python menentukan bagaimana instance kelas akan berperilaku.

```python
bilangan_bulat = 12345678910
bilangan_desimal = 1.56

class Angka:
    pass

bilangan = Angka()

print("Tipe data dari bilangan_bulat = ", type(bilangan_bulat))
print("Tipe data dari bilangan_desimal = ", type(bilangan_desimal))
print("Tipe data dari bilangan = ", type(bilangan))
```

Output :

```
TypeError: ini_metaclass.__new__() missing 3 required posit
PS D:\Belajar Python> & C:/Users/LENOVO/AppData/Local/Progr
Tipe data dari bilangan_bulat =  <class 'int'>
Tipe data dari bilangan_desimal =  <class 'float'>
Tipe data dari bilangan =  <class '__main__.Angka'>
PS D:\Belajar Python>
```

Ilustrasi :



Metaclass dapat diterapkan dalam pemrograman dengan konsep yang cukup abstrak. Sehingga memerlukan pemahaman yang baik dalam menggunakan konsep metaclass.

KESIMPULAN

1. Apa itu interface dan kapan kita perlu memakainya?
   - Interface merupakan koleksi dari method/ fungsi yang perlu disediakan oleh implementing class (child class). Interface biasanya dipakai untuk menulis kode yang terorganisir.

2. Apa itu kelas abstrak dan kapan kita perlu memakainya? Apa perbedaannya dengan interface?
   - Kelas Abstrak merupakan sebuah kelas yang didalamnya terdapat satu atau lebih metode abstrak. Kelas Abstrak biasanya dipakai untuk membuat struktur logika penurunan di dalam pemrograman objek.
   - Yang membedakan kelas abstrak dengan interface yaitu, kelas abstrak mempunyai metode abstrak dan konkrit. Kelas abstrak memiliki proses yang relative lebih cepat dari interface.

3. Apa itu kelas konkret dan kapan kita perlu memakainya?
   - Kelas konkret adalah turunan dari kelas abstrak yang biasanya dipakai untuk membuat implementasi dari kelas abstrak yang sebelumnya telah didefinisikan.

4. Apa itu metaclass dan kapan kita perlu memakainya? Apa bedanya dengan inheritance biasa?
   - Metaclass adalah kelas dari kelas yang mendefinisikan bagaimana kelas berperilaku. Metaclass biasanya diterapkan dalam pemrograman dengan konsep yang cukup abstrak.
   - Perbedaan metaclass dan inheritance biasa yaitu, metaclass mendefinisikan sebuah kelas, sedangkan inheritance adalah class yang dapat menurunkan atributnya untuk kelas turunannya.

DAFTAR PUSTAKA

- https://www.geeksforgeeks.org/abstract-classes-in-python/
- https://realpython.com/python-interface/
- https://www.datacamp.com/tutorial/python-metaclasses#rdl
- https://docs.python.org/id/3.11/contents.html