RESUME TUGAS 6
LAPORAN PRAKTIKUM

PEMPROGRAMAN BERORIENTASI OBJEK

Oleh :

Nama : Hafiza Eka Ramadhini

NIM : 121140048

Kelas : PBO RB

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

LAMPUNG SELATAN

2023

RINGKASAN

A. KELAS ABSTRAK

Kelas Abstrak atau Abstract Class adalah kelas yang tidak bisa diinstansiasi dan biasanya digunakan sebagai kerangka kerja untuk kelas turunan. Abstract Class digunakan untuk memaksa implementasi dari sebuah method yang harus ada di kelas turunan. Untuk membuat kelas abstrak di Python, Anda perlu mengimpor modul abc dan menggunakan kelas ABC sebagai kelas dasar. Anda juga perlu mendekorasi metode abstrak menggunakan dekorator @abstractmethod.

Contoh program sederhana yang menunjukkan penggunaan kelas abstrak di Python:

```
from abc import ABC, abstractmethod

class Bentuk(ABC):
    @abstractmethod
    def hitung_luas(self):
        pass

class PersegiPanjang(Bentuk):
    def __init_(self, panjang, lebar):
        self.panjang = panjang
        self.lebar = lebar

    def hitung_luas(self):
        return self.panjang * self.lebar

class Lingkaran(Bentuk):
    def __init_(self, jari_jari):
        self.jari_jari = jari_jari

    def hitung_luas(self):
        return 3.14 * self.jari_jari ** 2

persegi_panjang = PersegiPanjang(4, 5)
lingkaran = Lingkaran(3)

print(persegi_panjang.hitung_luas())
print(lingkaran.hitung_luas())
```

Dalam contoh program di atas, kita mendefinisikan kelas Bentuk sebagai kelas abstrak yang memiliki satu metode abstrak yaitu hitung_luas(). Metode ini harus diimplementasikan oleh kelas turunan. Kemudian kita mendefinisikan kelas PersegiPanjang dan Lingkaran sebagai kelas turunan dari kelas Bentuk. Kedua kelas ini harus mengimplementasikan metode hitung_luas().

Dalam program utama, kita membuat objek PersegiPanjang dan Lingkaran dan memanggil metode hitung_luas(). Metode ini akan mengembalikan luas dari masing-masing bentuk, yang dihitung berdasarkan implementasi di dalam kelas turunan.

B. INTERFACE

Dalam pemrograman berorientasi objek, sebuah Interface merupakan sebuah kesepakatan yang berisi definisi-definisi metode tanpa implementasi yang ada pada kelas interface itu sendiri. Implementasi dari metode-metode tersebut harus dilakukan pada kelas yang menggunakan interface tersebut.
Dalam Python, interface tidak secara eksplisit didukung, namun kita bisa membuat interface dengan menggunakan kelas abstrak dan decorator @abstractmethod dari modul abc.

Contoh program yang menunjukkan bagaimana membuat interface di Python menggunakan kelas abstrak dan decorator @abstractmethod:

```
from abc import ABC, abstractmethod

class Interface(ABC):
    @abstractmethod
    def method1(self):
        pass

    @abstractmethod
    def method2(self, arg):
        pass

class MyClass(Interface):
    def method1(self):
        print("Implementing method1")

    def method2(self, arg):
        print("Implementing method2 with argument: ", arg)

my_obj = MyClass()
my_obj.method1()
my_obj.method2("hello")
```

Pada contoh program di atas, kita telah membuat sebuah antarmuka (interface) yang diberi nama Interface yang memiliki dua metode abstrak, yaitu method1 dan method2. Kemudian, kita membuat sebuah kelas bernama MyClass yang mengimplementasikan kedua metode tersebut. Setelah itu, kita menciptakan sebuah objek dari kelas MyClass dan memanggil kedua metodenya.

Namun, jika kita tidak mengimplementasikan kedua metode abstrak tersebut pada kelas MyClass, maka akan terjadi kesalahan karena kelas tersebut tidak memenuhi persyaratan yang telah ditetapkan oleh antarmuka (interface) Interface.

C. METACLASS

Metaclass adalah sebuah kelas yang digunakan untuk membuat kelas baru. Dengan menggunakan metaclass, kita dapat mengubah perilaku pembuatan kelas dan menambahkan fungsi tambahan pada kelas-kelas tersebut secara fleksibel. Artinya, metaclass memberikan cara yang lebih terstruktur dan mudah untuk mengelola pembuatan kelas dan mengimplementasikan fitur-fitur tambahan pada kelas-kelas tersebut. Dengan demikian, metaclass memungkinkan kita untuk membuat kelas-kelas dengan perilaku dan fitur yang sesuai dengan kebutuhan kita.

Contoh program sederhana yang menunjukkan penggunaan metaclass:

```
class MyMeta(type):
    def __new__(cls, name, bases, attrs):
        attrs['my_var'] = 10
        return super().__new__(cls, name, bases, attrs)

class MyClass(metaclass=MyMeta):
    pass

obj = MyClass()
print(obj.my_var)  # Output: 10
```

Pada contoh program di atas, metaclass MyMeta menambahkan atribut my_class_var dan metode my_method pada kelas MyClass. Selain itu, metaclass MyMeta juga mengeksekusi method_init_ketika kelas baru dibuat. Kita membuat instance dari kelas MyClass dan memanggil metode dan atribut yang ditambahkan oleh metaclass MyMeta.

Metaclass bernama MyMeta yang bertugas mengontrol pembuatan kelas baru. MyMeta ini mengontrol atribut dan metode yang akan dimiliki oleh kelas baru yang akan dihasilkan. Selanjutnya, sebuah kelas baru bernama MyClass dibuat dengan menggunakan metaclass MyMeta. Karena metaclass MyMeta telah diatur dengan cara tertentu, maka kelas MyClass memiliki atribut dan metode yang berbeda dari kelas biasa.

KESIMPULAN

1. Apa itu interface dan kapan kita perlu memakainya?

   Jawab:
   Interface adalah sebuah kontrak yang digunakan untuk menentukan perilaku kelas-kelas yang mengimplementasikan kontrak tersebut. Interface berguna ketika kita ingin memastikan bahwa suatu kelas mempunyai metode tertentu dengan nama dan parameter yang sudah ditentukan.

2. Apa itu kelas abstrak dan kapan kita perlu memakainya? Apa perbedaannya dengan interface?

   Jawab:
   Kelas abstrak adalah kelas yang tidak bisa di-instantiate secara langsung, tetapi hanya bisa digunakan sebagai superclass untuk kelas-kelas turunannya. Kelas abstrak umumnya digunakan ketika kita ingin memastikan bahwa kelas turunan mengimplementasikan metode tertentu. Perbedaannya dengan interface adalah bahwa kelas abstrak dapat memiliki implementasi method yang tidak seluruhnya harus diimplementasikan oleh kelas turunan.

3. Apa itu kelas konkret dan kapan kita perlu memakainya?

   Jawab:
   Kelas konkret adalah kelas yang bisa di-instantiate langsung. Kelas konkret digunakan ketika kita ingin membuat objek-objek yang bisa langsung digunakan tanpa harus membuat turunan dari kelas tersebut.

4. Apa itu metaclass dan kapan kita perlu memakainya? Apa bedanya dengan inheritance biasa?

   Jawab:
   Metaclass berguna ketika kita ingin mengubah perilaku pembuatan kelas dan menambahkan fungsi tambahan pada kelas-kelas tersebut. Perbedaan antara metaclass dengan inheritance biasa adalah bahwa inheritance hanya memungkinkan kelas turunan untuk mengakses properti dan metode dari kelas induk, sedangkan metaclass memungkinkan pengontrolan lebih lanjut pada pembuatan kelas-kelas baru.

DAFTAR PUSTAKA

GeeksforGeeks. (n.d.). Abstract Classes in Python. from https://www.geeksforgeeks.org/abstract-classes-in-python/

van Rossum, G., & Warsaw, B. (2007). PEP 3115 -- Metaclasses in Python 3000. Python Software Foundation, from https://legacy.python.org/dev/peps/pep-3115/