LAPORAN MODUL 6

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Disusun oleh:

Nama: Riksan Cahyowadi

NIM: 121140106

Kelas : PBO (RB)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

1. **Kelas Abstrak**

Kelas abstrak adalah kelas dalam pemrograman berorientasi objek yang tidak dapat diinstansiasi. Namun, kelas abstrak dapat digunakan sebagai blueprint untuk kelas turunannya. Di Python, meskipun tidak secara langsung mendukung kelas abstrak, modul "abc" dapat digunakan untuk mendefinisikan kelas dasar abstrak. Penggunaan kelas abstrak dapat membantu berbagi kode di antara beberapa kelas yang terkait erat. Metode abstrak dapat didefinisikan dengan menggunakan dekorator @abstractmethod(), yang dapat digunakan untuk mendeklarasikan metode abstrak untuk property dan descriptor. Jika sebuah kelas abstrak mengandung metode abstrak dan kelas tersebut diwariskan ke kelas turunannya, maka metode tersebut harus dideklarasikan ulang (overriding method) dengan menambahkan pernyataan dalam tubuh metodenya.. Agar dapat beroperasi dengan benar dengan mesin kelas abstrak, descriptor harus mengidentifikasi dirinya sebagai abstrak menggunakan isabtrarctmethod. Berikut adalah contoh kode kelas abstrak:

```python
from abc import ABC, abstractmethod

class FiturGPS(ABC):
    @abstractmethod
    def get_lokasi(self):
        pass

    def fitur_baru(self):
        print("Tes fitur baru...")

class VoiceAssitance(ABC):
    @abstractmethod
    def aktifkan_asisten(self):
        pass

```

2. Interface

Interface adalah cara menulis kode terorganisir dan untuk mencapai abstraksi. Interface bertindak sebagai blueprint untuk merancang kelas, sehingga interface diimplementasikan di kelas. Jika sebuah kelas mengimplementasikan interface, maka instance dari kelas menyediakan interfacenya. Objek dapat menyediakan antarmuka secara langsung, selain apa yang diterapkan oleh kelas mereka.Karena metode abstrak adalah metode yang didefinisikan oleh interface, ini dilakukan oleh kelas-kelas, yang kemudian mengimplementasikan interface.

Pada tingkat yang lebih tinggi, interface berfungsi sebagai panduan untuk merancang kelas. Seperti kelas, antarmuka menetapkan metode, tetapi metode dalam antarmuka adalah metode abstrak. Metode abstrak didefinisikan oleh interface tetapi tidak memiliki implementasi konkret. Implementasi ini dilakukan oleh kelas-kelas yang mengimplementasikan interface dan memberikan implementasi konkret pada metode abstrak yang ada dalam antarmuka tersebut.

Terdapat 2 jenis interface pada python yaitu:

a. Informal interface, merupakan kelas yang mendefinisikan metode yang dapat diganti. Informal interface tidak memiliki deklarasi resmi atau aturan ketat yang harus diikuti, tetapi hanya berdasarkan kesepakatan atau konvensi yang dipahami oleh para pengembang yang terlibat dalam proyek tersebut.
b. Formal interface, merupakan antarmuka yang didefinisikan secara eksplisit dengan menggunakan mekanisme bahasa pemrograman yang didukung (seperti abc.ABCMeta) .Memerlukan implementasi lengkap dari metode atau atribut yang didefinisikan dalam antarmuka tersebut.

Contoh kode interface:

```python
from abc import ABC, abstractmethod

class InterfaceRiksan(ABC):
    @abstractmethod
    def tes(self):
        pass

class MyClass(InterfaceRiksan):
    pass

class MyClass(InterfaceRiksan):
    def tes(self):
        print("Nama saya Riksan Cahyowadi")

cetak1 = MyClass()
cetak1.tes()
```

3. Metaclass

Dalam bahasa pemrograman Python, metaclass adalah suatu kelas yang digunakan untuk menciptakan dan mengatur kelas-kelas lain. Metaclass berfungsi sebagai "pembuat kelas" atau "pengatur kelas", yang dapat mengendalikan cara kelas-kelas lain dibuat, diinisialisasi, atau dikelola. Ada 2 pendekatan metaclass, yaitu:

- Kelas gaya lama (diperkenalkan sebelum python versi 2.2)
  Dengan kelas gaya lama, kelas dan tipe bukanlah hal yang persis sama. Instance dari kelas gaya lama selalu diimplementasikan dari satu tipe bawaan yang disebut instance.

- Kelas gaya baru (diperkenalkan pada python versi 2.2)
  Kelas gaya baru menyatukan konsep kelas dan tipe. Jika *obj* merupakan turunan dari kelas gaya baru, *type(obj)* sama dengan *obj.__class__:.*

Dalam Python 3, semua kelas dianggap sebagai kelas gaya baru. Oleh karena itu, adalah wajar untuk menggunakan istilah "tipe objek" dan "kelas" secara bergantian dalam konteks Python 3.

Contoh metaclass:

```
1   class CekAngka(type):
2       def __init__(cls, X, Y, Z ):
3
4           cls.value = 76
5
6   class X(metaclass=CekAngka):
7       pass
8
9   X.value
10
11
12  class Y(metaclass=CekAngka):
13      pass
14
15  Y.value
16
17
18  class Z(metaclass=CekAngka):
19      pass
20
21  Z.value
```

4. Kesimpulan

• Interface adalah cara menulis kode terorganisir dan untuk mencapai abstraksi. Interface bertindak sebagai blueprint untuk merancang kelas, sehingga interface diimplementasikan di kelas. Penggunaan interface atau informal interface cocok untuk proyek kecil dengan sedikit pengembang yang mengerjakan kode sumber. Namun, ketika proyek semakin besar dan tim bertambah, hal ini bisa mengakibatkan pengembang menghabiskan banyak waktu untuk mencari kesalahan logika yang sulit ditemukan dalam basis kode.

• Kelas abstrak adalah kelas yang tidak bisa diinstansiasi, tetapi bisa digunakan untuk membuat kelas turunannya. Pembuatan kelas abstrak biasanya bertujuan untuk membuat blueprint untuk kelas turunannya. Penggunaan kelas abstrak berguna saat metode atau atribut dalam kode tidak ingin ditampilkan kepada pengguna, hanya menampilkan program utamanya. Perbedaan antara kelas abstrak dan interface terletak pada fakta bahwa kelas abstrak bisa memiliki metode abstrak maupun metode konkret, sedangkan pada interface semua metodenya bersifat abstrak.

• Kelas konkrit merupakan kelas yang bukan kelas abstrak, intinya dalam pengimplementasiannya tidak menggunakan ABC (Abstract Base Class). Untuk waktu kapan penggunaanya dapat digunakan saat kita ingin membuat blueprint untuk kelas turunan.

• Metaclass adalah suatu kelas yang digunakan untuk menciptakan dan mengatur kelas-kelas lain. Metaclass berfungsi sebagai "pembuat kelas" atau "pengatur kelas", yang dapat mengendalikan cara kelas-kelas lain dibuat, diinisialisasi, atau dikelola. Penggunaan metaclass digunakan untuk menentukan tipe data dari kelas tersebut sehingga tipe datanya tidak akan berubah. Dapat dikatakan, perbedaan yang mencolok antara Metaclass dan Inheritance adalah metaclass merupakan tipe data khusus untuk membuat kelas, sedangkan inheritance merupakan mekanisme dalam OOP untuk mewariskan kelas.

DAFTAR PUSTAKA

Praktikum PBO Modul 6

https://www.teachmesoft.com/2020/02/bab-8-kelas-abstrak-abstract-class.html

https://realpython.com/python-interface/

https://www.datacamp.com/tutorial/python-metaclasses

https://realpython.com/python-metaclasses/

https://www.geeksforgeeks.org/abstract-classes-in-python/