TUGAS PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

KELAS ABSTRAK, INTERFACE, DAN METACLASS

Oleh :

Vebie Yoseva Theresia Pasaribu 121140016

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK ELEKTRO, INFORMATIKA DAN SISTEM FISIS

INSTITUT TEKNOLOGI SUMATERA

2023

PENJELASAN

1. KELAS ABSTRAK

Abstraksi dalam python dapat diterapkan melalui kelas yang dibuat. Abstraksi berguna untuk mengurangi kompleksitas dengan hanya memperlihatkan atribut dan menyembunyikan detail yang kurang penting dari pengguna. Dengan itu, pengguna hanya mengetahi apa yang dikerjakan oleh obkjek, tetapi tidak mengetahui seperti apa mekanisme yang terjadi dibelakangnya.

Dalam PBO, objek yang dibuat tidak boleh bersifat abstrak atau hanya dapat dibayangkan saja. Contoh objek yang bersifat abstrak adalah seperti makanan, makhluk hidup, permainan, karya, dan lainnya. Sehingga untuk menggunakannya diperlukan kelas abstrak, dimana kita tidak dapat membuat objek dari kelas ini. Kelas abstrak mempunyai setidaknya satu atau lebih fungsi yang bersifat abstrak dan untuk menggunakan kelas ini, dapat dengan menerapkan pewarisan kelas abstrak sebagai kelas super atau superclass. Kelas turunan atau subkelas yang mewarisi kelas abstrak harus mengimplementasikan (override) dari semua fungsi abstrak yang didefenisikan kelas abstrak tersebut.

Dalam kelas abstrak juga masih dapat mendefenisikan fungsi biasa dan dapat mendefenisikan konstruktor yang akan dipanggil dari kelas turnan yang mewarisi kelas abstrak. Kelas abstrak ditandai dengan adanya pewarisan pada kelas ABC atau *Abstract Base Class* dan dapat juga ditandai dengan decorator @abstractmethod pada fungsinya. Kelas ABC ini yang akan membantu mendefenisikan kelas abstrak.

```
from abc import ABC, abstractmethod #menggunakan kelas ABC


class Abstractclass(ABC):
    @abstractmethod #menggunakan dekorator @abstractmethod
    def abstract_method(self):
        pass
```

Contoh code :

```
from abc import ABC, abstractmethod

class Bangundatar(ABC):
    @abstractmethod
    def keliling(self):
        return 2 * 3.14 * self.jejari

    @abstractmethod
    def luas(self):
        return 3.14 * self.jejari * self.jejari

class Lingkaran(Bangundatar):
    def __init__(self, jejari):
        self.jejari = jejari

    def keliling(self):
        return 2 * 3.14 * self.jejari

    def luas(self):
        return 3.14 * self.jejari * self.jejari

bangun = Lingkaran(10)
print(bangun.luas())
print(bangun.keliling())
```

2. INTERFACE

Interface merupakan sebuah blok yang berisi kumpulan method tanpa tubh yabng mendefensikan method umum yang dapat menghubungkan kelas- kelas yang berbeda. Dengan interface dapat mendefenisikan pekerjaan sebuah kelas tanpa harus melihat terlebih dahulu implementasinya. Jadi dapat dikatakan interface memungkinkan pemrogram dalam mengiplementasikan method yang sama terdahap kelas yang tidak memiliki hubungan. Interface ini mempermudah dalam desain program skala besar. Interface sedikit mirip dengan kelas abstrak, tetapi seluruh fungsi atau method dalam interface didefenisikan sebagi abstrak dan harus bersifat public. Interface biasanya kecepatan prosesnya relative lebih lambat dari kelas abstrak.

Contoh :

```
from abc import ABC, abstractmethod

class FiturBT(ABC):
    @abstractmethod
    def pair(self, other):
        self.other = other
        if other is not None :
            print("Terhubung!")
        else :
            print("Gagal!")

    @abstractmethod
    def unpair(self):
         if self.other is not None:
            print("Terputus!")
         else :
            print("Tidak ada perangkat yang terhubung!")

class FiturVA(ABC):
    @abstractmethod
    def aktifkan_VA(self):
        status = "Voice Assistance diaktifkan !"
        print(status)

    @abstractmethod
    def matikan_VA(self):
        status = "Voice Assistance dimatikan !"
        print(status)

    @abstractmethod
    def greetings(self):
        user = input("Masukan username : ")
        print("Halo ", user, " Selamat datang :)")

class Smarthphone(FiturBT, FiturVA):

    def pair(self, other=None):
        self.other = other
        if other is not None :
            print("Terhubung!")
        else :
            print("Gagal!")

    def unpair(self):
        if self.other is not None:
            print("Terputus!")
        else :
            print("Tidak ada perangkat yang terhubung!")

    def aktifkan_VA(self):
        status = "Voice Assistance diaktifkan !"
        print(status)
    def matikan_VA(self):
        status = "Voice Assistance dimatikan !"
        print(status)
    def greetings(self):
        user = input("masukan username : ")
        print("Halo ", user, " Selamat datang :)")

hp = Smarthphone()
hp.pair("yoseva")
hp.unpair()
hp.greetings()
hp.aktifkan_VA()
hp.matikan_VA()
```

3. METACLASS

Semua kelas- kelas yang ada dalam python merupakan bagian dari metaclass. Metaclass digunakan untuk membuat kelas baru serta mengatur bagaimana perilaku dan sifat yang dimiliki kelas dalam python yang dibuat. Metaclass dapat dibuat dengan 2 cara, yaitu menggunakan ‘ __new__’ method yang membuat dan mengembalikan kelas objek yang baru dan ‘__init__’ yaitu method yang akan menginisialisasi objek yang baru dibuat. Metaclass juga dapat memberikan fleksibilitas dan kustomisasi kelas. Metaclass juga memberikan kelas turunan melalui pewarisan dan melakukan komversi method menjadi statis untuk menghasilakn pengotimalan yang lebih baik.

KESIMPULAN

Kelas abstrak merupakan kelas yang berguna untuk mengurangi kompleksitas dengan hanya memperlihatkan atribut dan menyembunyikan detail yang kurang penting dari pengguna. Kelas abstrak mempunyai setidaknya satu atau lebih fungsi yang bersifat abstrak dan untuk menggunakan kelas ini, dapat dengan menerapkan pewarisan kelas abstrak sebagai kelas super atau superclass. Kelas abstrak digunakan saat ingin mengimplementasikan method atau membangun kontrak atau interface agar kelas-kelas turunan yang dibuat mengikuti perilaku tertentu.

Interface merupakan sebuah blok yang berisi kumpulan method tanpa tubh yabng mendefensikan method umum yang dapat menghubungkan kelas- kelas yang berbeda. Dengan interface dapat mendefenisikan pekerjaan sebuah kelas tanpa harus melihat terlebih dahulu implementasinya. Interface biasanya digunakan untuk mendefenisikan kontrak atau perilaku yang dilakukan sebuah kelas turunan.

Kelas abstrak dan interface merupakan dua hal yang sedikit mirip, akan tetapi mereka memiliki perbedaan seperti : kelas abstrak memiliki decorator @abstractmethod dalam method abstrak yang harus diimplementasikan pada kelas turunan. Sedangkan interface dapat dibentuk dari adanya penerapan kelas abstrak. Kelas abstrak memiliki implementasi method umum yang dapat diwarisi kelas turunan. Sedangkan interface, tidak memiliki implementasi method.

Dalam python, setiap objek memiliki tipe data. Kelas yang dibuat masih dapat memunkinkan untuk membuat kelas turunan dari kelas tersebut. Akan tetapi, kelas juga merupakan sebuah objek. Metaclass merupakan kelas dari suatu kelas. Metaclass berguna untuk membuat atau menurunkan kelas baru. Dengan medefenisikan metaclass akan dapat menyesuaikan hal- hal seperti perilaku dan sifat dari kelas atau method yang baru dibuat.

REFERENSI

Modul Praktikum Pemrograman Berorientasi Objek
https://www.geeksforgeeks.org/abstract-classes-in-python/
https://www.pythontutorial.net/python-oop/python-abstract-class/
https://auftechnique.com/4-pillar-pemrograman-berorientasi-objek/
https://www.teachmesoft.com/2020/02/bab-8-kelas-abstrak-abstract-class.html
https://www.algonina.com/blog/tutorial-typescript-abstract-class-dan-interface
https://idcsharp.com/2019/12/09/interface-dan-abstract-class-apa-perbedaannya/
https://realpython.com/python-metaclasses/
https://www.pythontutorial.net/python-oop/python-metaclass/
https://www.youtube.com/watch?v=00acMGEfVlo
https://www.youtube.com/watch?v=yWzMiaqnpkI