LAPORAN MODUL 6

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Disusun oleh:

Nama : Dimas Saputra

NIM : 121140059

Kelas : PBO RB

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2023

1. **Kelas Abstrak**

Dalam pemrograman berorientasi objek, kelas abstrak adalah kelas yang tidak dapat dibuat instance-nya. Namun, bisa membuat kelas yang mewarisi dari kelas abstrak. Pembuatan kelas abstrak biasanya untuk membuat blueprint untuk kelas turunannya. Dalam python tidak secara langsung mendukung kelas abstrak. Tapi python tetap menawarkan modul yang memungkinkan untuk mendefinisikan kelas abstrak. Untuk mendefinisikan kelas abstrak dapat menggunakan modul "abc" (abstract base class), Modul ini digunakan dalam infrastruktur untuk mendefinisikan kelas dasar abstrak. Dalam prakteknya, dalam penggunaan kelas abstrak untuk berbagi kode di antara beberapa kelas yang terkait erat.

Untuk mendefinisikan metode abstrak dapat menggunakan dekorator *@abstractmethod()*. *@abstractmethod()* dapat digunakan untuk mendeklarasikan metode abstrak untuk property dan descriptor. Penggunaan dekorator ini untuk pemanggilan metode abstrak yang digunakan pada kelas turunannya. Metode abstrak dapat dipanggil menggunakan satu mekanisme panggilan "super" normal. Ketika *@abstractmethod()* diterapkan dalam kombinasi dengan descriptor metode lain, itu harus diterapkan sebagai decorator terdalam. Agar dapat beroperasi dengan benar dengan mesin kelas abstrak, descriptor harus mengidentifikasi dirinya sebagai abstrak menggunakan __isabtrarctmethod__.

Contoh kode kelas abstrak:

```python
from abc import ABC, abstractmethod

class Biodata(ABC):
    @abstractmethod
    def __init__(self,nama,nim):
        pass

    @abstractmethod
    def cetak(self):
        pass

class Mahasiswa(Biodata):
    def __init__(self,nama,nim):
        self.nama = nama
        self.nim = nim

    def cetak(self):
        print(f"Nama mahasiswa : {self.nama}")
        print(f"NIM mahasiswa : {self.nim}")
```

2. **Interface**

Menerapkan 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.

Informal Interface

Sifat dinamis python memungkinkan untuk mengimplementasikan informala interface. Informal interface python adalah kelas yang mendefinisikan metode yang dapat diganti, tetapi tidak ada penerapan yang ketat. Informal interface sangat bagus untuk proyek kecil dimana hanya beberapa pengembang yang mengerjakan kode sumber. Namun, saat proyek semakin besar dan tim bertambah, hal ini dapat menyebabkan pengembang menghabiskan waktu berjam-jam untuk mencari kesalahan logika yang sulit ditemukan di basis kode. Interface dapat memperluas interface lain dengan mencantumkan interface lain sebagai interface dasar.

Formal Interface

Pada formal interface kelas abstrak dapat dibangun hanya dengan sedikit baris kode, untuk kemudian diimplementasikan pada kelas turunannya. Dan implementasi ini bersifat wajib sehingga dinamakan formal interface

Contoh kode interface:

```python
from abc import ABC, abstractmethod

class Charger(ABC):
    @abstractmethod
    def print(self):
        pass

class Laptop(Charger):
    def __init__(self, sisa_battery):
        self.sisa_battery = sisa_battery

    def print(self):
        print(f"Power battery sisa {self.sisa_battery}% sedang terisi")
```

3. **Metaclass**

Metaclass adalah konsep OOP esoteris, yaitu bersembunyi dibalik hampir semua kode python. Jadi metaclasss adalah tipe kelas special yang berfungsi untuk membuat kelas-kelas pada python.

- Kelas gaya lama

  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. Jika *obj* adalah
- Kelas gaya baru

  Kelas gaya baru menyatukan konsep kelas dan tipe. Jika *obj* merupakan turunan dari kelas gaya baru, *type(obj)* sama dengan *obj.__class__:*.

Karena di python 3, semua kelas adalah kelas gaya baru, jadi untuk merujuk ke tipe objek dan kelasnya secara bergantian. Pada python semuanya adalah objek, kelas adalah objek juga, akibatnya kelas harus memiliki tipe. *type* adalah metaclass, yang kelasnya adalah instance.

Contoh kode metaclass:

```
class Model3D(type):
    def __init__(self, panjang):
        self.panjang = panjang

class X(metaclass=Model3D):
    pass

class Y(metaclass=Model3D):
    pass

class Z(metaclass=Model3D):
    pass
```

4. Kesimpulan

- Interface adalah cara menulis kode terorganisir dan untuk mencapai abstraksi. Interface bertindak sebagai blueprint untuk merancang sebuah kelas, sehingga interface diimplementasikan di dalam kelas. Penggunaan interface atau yang informal interface, sangat bagus digunakan untuk proyek kecil dimana hanya beberapa pengembang yang mengerjakan kode sumber. Namun, saat proyek semakin besar dan tim bertambah, hal ini dapat menyebabkan pengembang menghabiskan waktu berjam-jam untuk mencari kesalahan logika yang sulit ditemukan di basis kode.
- kelas abstrak adalah kelas yang tidak dapat dibuat instance-nya. Namun, bisa membuat kelas yang mewarisi dari kelas abstrak. Pembuatan kelas abstrak biasanya untuk membuat blueprint untuk kelas turunannya. Penggunaan kelas abstrak saat metode atau atribut yang ada pada kode tidak ingin ditampilkan pada penggunanya, jadi hanya menampilkan main programnya. Perbedaan antara kelas abstrak dan interface ada pada: kelas abstrak dapat mempunyai metode abstrak atau metode konkrit, sedangkan interface semua metode bersifat abstrak.
- Kelas konkrit adalah kelas yang bukan kelas abstrak yang berarti penggunaannya tidak menggunakan ABC (Abstract Base Class) dalam implementasinya, tapi digunakan untuk mendefinisikan kelas abstrak atau dengan kata lain, kelas konkrit sama dengan kelas turunan kelas abstrak. Penggunaan kelas konkrit untuk menggunakan metode abstrak dari kelas abstrak agar dapat dibuat instancenya.
- Metaclass adalah konsep OOP esoteris, yaitu bersembunyi dibalik hampir semua kode python. Jadi metaclasss adalah tipe kelas special yang berfungsi untuk membuat kelas-kelas pada python. Penggunaan metaclass untuk menentukan tipe data dari kelas tersebut sehingga tipe data menjadi tetap. Perbedaan dari metaclass dan interface, pada metaclass kelas adalah objek, sedangkan interface kelas dan objek kedua hal yang berbeda.

DAFTAR PUSTAKA

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

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

https://www.geeksforgeeks.org/python-interface-module/

https://www.pythontutorial.net/python-oop/python-abstract-class/

https://docs.python.org/3/library/abc.html

Modul 6 Praktikum PBO